

FINAL  
IN-63-CR

201250  
178 P

## A Reliable Algorithm for Optimal Control Synthesis

Brett VanSteenwyk and Uy-Loi Ly  
Department of Aeronautics and Astronautics, FS-10  
University of Washington  
Seattle, WA 98195

### FINAL TECHNICAL REPORT

Supported under NASA Ames Research Grant Contract NAG-2 - 691  
Period: January 1991 to December 1992

(NASA-CR-194809) A RELIABLE  
ALGORITHM FOR OPTIMAL CONTROL  
SYNTHESIS Final Technical Report,  
Jan. 1991 - Dec. 1992 (Washington  
Univ.) 178 p

N94-23332

Unclass

G3/63 0201250

## ACKNOWLEDGEMENTS

This research was supported in part by NASA Ames Research Center under Grant NAG-2-691. The helpful comments and discussions with Dr. Marc Takahashi from NASA Ames, and his effort in providing us the dynamic models of the UH-60 rotorcraft in this study are gratefully acknowledged.

## Abstract

In recent years, powerful design tools for linear time-invariant multivariable control systems have been developed based on direct parameter optimization. In this report, an algorithm for reliable optimal control synthesis using parameter optimization is presented. Specifically, a robust numerical algorithm is developed for the evaluation of the  $H^2$ -like cost functional and its gradients with respect to the controller design parameters. The method is specifically designed to handle defective degenerate systems and is based on the well-known Padé series approximation of the matrix exponential. Numerical test problems in control synthesis for simple mechanical systems and for a flexible structure with densely packed modes illustrate positively the reliability of this method when compared to a method based on diagonalization.

Several types of cost functions have been considered: a cost function for robust control consisting of a linear combination of quadratic objectives for deterministic and random disturbances, and one representing an upper bound on the quadratic objective for worst-case initial conditions.

Finally, a framework for multivariable control synthesis has been developed combining the concept of closed-loop transfer recovery with numerical parameter optimization. The procedure enables designers to synthesize not only observer-based controllers but also controllers of arbitrary order and structure. Numerical design solutions rely heavily on the robust algorithm due to the high order of the synthesis model and the presence of near-overlapping modes. The design approach is successfully applied to the design of a high-bandwidth control system for a rotorcraft.

# Contents

<b>Glossary</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Optimal Control Synthesis and Parameter Optimization . . . . .	1
1.2 Key Research Motivation . . . . .	2
1.3 Summary of Contributions . . . . .	3
<b>2 Control Design Using Numerical Optimization</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Synthesis Model Description . . . . .	5
2.3 Formulation of the Closed-Loop System . . . . .	8
2.3.1 Case $D_c D_{su}^i = 0$ . . . . .	8
2.3.2 Case $D_{su}^i D_c = 0$ . . . . .	9
2.4 Design Cost Functions for $H^2$ Optimal Control . . . . .	9
2.4.1 Random Impulsive Disturbances . . . . .	10
2.4.2 Random White-Noise Disturbances . . . . .	10
2.4.3 Worst-Case Impulsive Disturbances . . . . .	12
2.4.4 Worst-Case for White-Noise Disturbances . . . . .	13
2.4.5 Design Features of Direct Optimization . . . . .	14
2.5 Derivation of Cost Function Gradients . . . . .	15
2.5.1 Gradients of $J_1(t_f, C_o)$ (Case $D_{su} D_c = 0$ ) . . . . .	16
2.5.2 Gradients of $J_1(t_f, C_o)$ (Case $D_c D_{su} = 0$ ) . . . . .	16
2.5.3 Gradients of $\tilde{J}_\sigma(t_f, C_o)$ . . . . .	17
2.5.4 Case $D_{su} D_c = 0$ . . . . .	17
2.5.5 Case $D_c D_{su} = 0$ . . . . .	17
2.6 Special Design Problems . . . . .	17
2.7 Design Example Using a Simplified Helicopter Model . . . . .	20
2.7.1 Helicopter Model . . . . .	21
2.7.2 Full-State Feedback Design . . . . .	21
2.7.3 Comparison of Output-Feedback Designs . . . . .	22
2.7.4 Actuator Disturbance Augmentation . . . . .	22
2.7.5 Fictitious State Noise Augmentation . . . . .	24
2.8 Conclusions . . . . .	24
<b>3 Evaluating <math>\mathcal{X}(t)</math> and <math>\mathcal{M}(t)</math></b>	<b>33</b>
3.1 Introduction . . . . .	33
3.2 Alternative Approaches for Solving $\mathcal{X}(t)$ and $\mathcal{M}(t)$ . . . . .	34
3.3 Matrix Exponential Approach . . . . .	35

3.4	Numerical Method for the Matrix Exponential . . . . .	36
3.5	Preconditioning With Scaling and Rotation . . . . .	38
3.6	Detailed Algorithm for Computing $\mathcal{X}(t)$ . . . . .	39
3.7	Detailed Algorithm for Computing $\mathcal{M}(t)$ . . . . .	40
3.8	Basic Test Results . . . . .	43
3.9	A Two-Mass-Spring Design Problem . . . . .	44
3.10	Degeneracy During Optimization . . . . .	47
3.11	Conclusions . . . . .	47
<b>4</b>	<b>Control Design for a Large Space Structure</b>	<b>49</b>
4.1	Introduction . . . . .	49
4.2	JPL Large Space Structure . . . . .	49
4.3	Controller Design . . . . .	51
<b>5</b>	<b>Closed-Loop Transfer Recovery</b>	<b>57</b>
5.1	Introduction . . . . .	57
5.2	Analysis of Closed-Loop Transfer Recovery . . . . .	57
5.3	The Special Coordinate Basis . . . . .	58
5.4	Conditions for Exact and Asymptotic Closed-Loop Transfer Recovery . . .	60
5.5	CLTR Design with a Full-Order Observer-Based Controller . . . . .	60
5.6	CLTR with a Luenberger Observer . . . . .	62
5.7	Conclusions . . . . .	64
<b>6</b>	<b>CLTR Using Numerical Optimization</b>	<b>65</b>
6.1	Introduction . . . . .	65
6.2	Problem Formulation . . . . .	66
6.3	Feedforward Controller in the CLTR Design Procedure . . . . .	68
6.3.1	A Feedforward Controller under State-Feedback . . . . .	68
6.3.2	A Feedforward Controller with Observer-Based Controllers . . . . .	70
6.3.3	A Feedforward Controller with Non-Observer-Based Controllers . .	72
6.4	Concluding Remarks . . . . .	74
<b>7</b>	<b>Control Design for a High-Performance Rotorcraft via CLTR</b>	<b>75</b>
7.1	Introduction . . . . .	75
7.2	The UH-60 Model . . . . .	76
7.3	Actuator and Sensor Delays . . . . .	77
7.4	ADS-33C Requirements and the Ideal Response Model . . . . .	77
7.5	State Feedback Designs . . . . .	79
7.6	Closed-Loop Transfer Recovery . . . . .	81
7.7	Luenberger Design . . . . .	81
7.8	Model Reduction on the Feedforward Controller . . . . .	84
7.9	Balanced Truncation of Luenberger Observer . . . . .	85
7.10	Reduction of Controller Order using Numerical Optimization . . . . .	86
7.11	Numerical CLTR . . . . .	87
7.12	Direct Optimization . . . . .	88
7.13	Conclusions . . . . .	90

<b>8</b>	<b>Future Work</b>	<b>101</b>
8.1	Evaluation of Worst-Case Method for Loop Transfer Recovery . . . . .	101
8.2	Completion of the Hybrid Algorithm . . . . .	101
8.3	Eigenvalue/Damping Constraints . . . . .	101
8.3.1	Current Method . . . . .	102
8.3.2	New Method of Eigenvalue Constraints . . . . .	103
<b>A</b>	<b>Preliminaries to the Hybrid Algorithm</b>	<b>109</b>
A.1	Introduction . . . . .	109
A.2	Converting a 2x2 block to $\sigma - \omega$ form. . . . .	109
A.3	Reduction of a 2x2 Block to Upper Triangular Form . . . . .	110
A.4	Reducing the Schur Matrix to the Diagonal Form . . . . .	111
A.5	Reducing the Upper Triangular Part Between 2 Real Roots . . . . .	112
A.6	Upper Triangular Part Between a Real Root and a Complex Pair . . . . .	112
A.7	Upper Triangular Part Between a Complex Pair and a Real Root . . . . .	113
A.8	Upper Triangular Block Between 2 Complex Pairs . . . . .	113
A.9	Using These Techniques for Zeroing Blocks . . . . .	113
<b>B</b>	<b>Cost and Gradient Computation Using The Hybrid Algorithm</b>	<b>115</b>
B.1	Introduction . . . . .	115
B.2	The $\mathcal{X}$ Integral . . . . .	116
B.3	The $\mathcal{M}$ Integral . . . . .	118
B.4	Padé Series for Matrix Exponential Integral . . . . .	120
B.5	Padé Series for Integral of Matrix Exponential and Sinusoid . . . . .	121
B.6	Scaling . . . . .	122
<b>C</b>	<b>An Algorithm for SCB</b>	<b>125</b>
C.1	S.C.B. for Non-Strictly Proper SISO Systems . . . . .	125
C.2	S.C.B. for Strictly Proper SISO Systems . . . . .	125
C.3	S.C.B. for Multi-Input and Multi-Output Systems . . . . .	129
C.4	SCB Transformations for Decomposing the Remaining States . . . . .	131
<b>D</b>	<b>Robust Gradient Routines</b>	<b>139</b>
<b>E</b>	<b>General Utility Routines for Synthesis and Analysis</b>	<b>145</b>
<b>F</b>	<b>UH-60 Rotocraft Design Files (Chapter 7)</b>	<b>151</b>
<b>G</b>	<b>UH-60 Rotocraft Models (Chapter 7)</b>	<b>173</b>
<b>H</b>	<b>JPL Large Space Structure Design Files (Chapter 4)</b>	<b>175</b>
<b>I</b>	<b>JPL Large Space Structure Model (Chapter 4)</b>	<b>177</b>
<b>J</b>	<b>One-Dimensional Rotocraft Design Files (Chapter 2.7.1)</b>	<b>179</b>

# List of Figures

2.1	Closed-Loop System with a Feedback/Feedforward Controller . . . . .	6
2.2	State-Feedback Design for the Helicopter Model . . . . .	27
2.3	$H^2$ Designs for Various Loop Recovery Weights . . . . .	28
2.4	Worst-Case Designs for Various Loop Recovery Weights . . . . .	29
2.5	$H^2$ Designs for Various Stability Weights . . . . .	30
2.6	Worst-Case Designs for Various Stability Weights . . . . .	31
3.1	A Two-Mass-Spring Mass System . . . . .	44
3.2	Behavior of Condition Number over the Entire Design Optimization . . . . .	46
4.1	Antenna Structure . . . . .	50
4.2	Hub Responses to Structure Excitation . . . . .	54
4.3	Hub Responses to Structure Excitation (cont.) . . . . .	55
5.1	S.C.B. Diagram . . . . .	59
6.1	State Feedback System with a Feedforward Gain . . . . .	69
6.2	Closed-Loop System with a Feedback/Feedforward Controller . . . . .	71
7.1	Synthesis Model for Full-State Design with Integral Controls . . . . .	79
7.2	Synthesis Model for Full-State Design with Sensor Delays Added . . . . .	80
7.3	Block Diagram for CLTR with Output-Feedback Controller $C(s)$ . . . . .	82
7.4	Synthesis Model for CLTR with Observer-Based Controllers . . . . .	82
7.5	Controller Design Structure in Direct Optimization . . . . .	89
7.6	Ideal UH-60 Command Responses . . . . .	95
7.7	Errors from Ideal Responses, Hover . . . . .	96
7.8	Errors from Ideal Responses, Hover . . . . .	97
7.9	Errors from Ideal Responses, Forward Flight . . . . .	98
7.10	Errors from Ideal Responses, Forward Flight . . . . .	99

# List of Tables

2.1	Static State Feedback for the Helicopter Model . . . . .	21
2.2	Traditional Loop Transfer Recovery Using Different Design Methods . . . .	23
2.3	Singular Value Type Disturbance Augmentation . . . . .	25
3.1	Closed-Loop Eigenvalues of the Two-Mass-Spring System . . . . .	45
4.1	Open Loop Modes of the Antenna Structure . . . . .	51
4.2	Design Variables: JPL Antenna Structure . . . . .	52
4.3	Boom-Dish-Controller Closed Loop Modes . . . . .	53
7.1	UH-60 Rotorcraft States . . . . .	76
7.2	Design Weights on the Criterion Variables in the LQ Synthesis . . . . .	80
7.3	Hankel Singular Values for Feedforward Controller Model . . . . .	85
7.4	Objective Weights in Numerical CLTR . . . . .	88
7.5	Objective Weights in Direct Optimization . . . . .	89
7.6	Hankel Singular Values, Luenberger Observer . . . . .	91
7.7	Single-Loop Sensor Robustness Properties, Hover . . . . .	91
7.8	Single-Loop Actuator Robustness Properties, Hover . . . . .	92
7.9	Multiloop Actuator Robustness Properties, Hover . . . . .	92
7.10	Single-Loop Sensor Robustness Properties, Forward Flight . . . . .	92
7.11	Single Loop Actuator Robustness Properties, Forward Flight . . . . .	93
7.12	Multiloop Actuator Robustness Properties, Forward Flight . . . . .	93
7.13	Design Evaluation, Hover . . . . .	93
7.14	Design Evaluation, Forward Flight . . . . .	94

# Glossary

<b>LQR</b>	Linear Quadratic Regulator
<b>CLTR</b>	Closed Loop Transfer Recovery
<b>SCB</b>	Special Coordinate Basis
$\Delta$	Change from nominal or error quantity
$\delta$	Variation
$tr \{*\}$	Trace of a matrix
$E[*]$	Expected value
$\bar{\sigma}$	Maximum singular value
$Q$	Criterion penalty matrix
$R$	Control penalty matrix
$t_f$	Final time
$u$	Control vector
$v$	Measurement disturbance vector
$w$	Process disturbance vector
$x$	Plant model states
$F$	Plant dynamics matrix
$G$	Plant control input distribution matrix
$\Gamma$	Plant disturbance input distribution matrix
$H_s$	Plant sensor output distribution matrix
$H_z$	Plant criterion output distribution matrix
$D_{su}$	Plant control input to sensor output direct feedthrough matrix
$D_{sw}$	Plant disturbance input to sensor output direct feedthrough matrix
$D_{zu}$	Plant control input to criterion output direct feedthrough matrix
$D_{zw}$	Plant disturbance input to criterion output direct feedthrough matrix
$y_s$	Sensor output
$z$	Criterion output
$x_c$	Controller states
$A_c$	Controller dynamics matrix
$B_c$	Controller input distribution matrix
$C_c$	Controller output distribution matrix
$D_c$	Controller direct feedthrough matrix

# Chapter 1

## Introduction

### 1.1 Optimal Control Synthesis and Parameter Optimization

Traditional design methods in linear optimal control for continuous-time systems have been extensively treated [1]. Development of these control systems is usually based on the characterization of the control problem under the setting of optimization of the two-norm of a set of controlled output responses to random disturbance inputs or initial conditions. Additional consideration of design robustness is taken by formulating the problem to include  $H^\infty$ -norm bound constraints for a class of additive and multiplicative uncertainties applied at the plant inputs and/or outputs. Solutions are obtained for both the state- and output-feedback design problems and involve, in the majority of cases, solving an appropriate set of algebraic Riccati equations [4, 5]. Theoretical studies of these approaches have been a major concern of researchers in the control field and a major breakthrough has been made in recent work by Stoorvogel [13, 14].

An alternate and less often mentioned design option for robust multivariable control is based on direct numerical optimization of a  $H^2$  performance objective with a given pre-specified controller structure. Early work in this area has been published by Levine and Athans [15], Anderson and Moore [16], and more recently, an extensive review of the subject was presented by Makila [17]. More recently, a new look into parameter optimization as it applies to multivariable control synthesis is provided by Ly [18] where he used a quadratic performance objective based on a finite-time horizon and the nonlinear optimization technique of Ref. [21].

Synthesis of robust multivariable control systems using nonlinear constrained optimization has become increasingly important in recent years [40]–[41] due to the design flexibility this approach offers. A numerical design package *SANDY* was developed to provide for the first time such an effective controller design tool. The plant and controller models are assumed to be linear and time invariant. The controller can possess a non-observer based structure and be of much lower order than the plant. To ensure good responses in critical spectral regions, a frequency-shaped  $H^2$  performance objective can be implemented by means of bandpass filters. Robustness can be enhanced by simultaneously optimizing over several plant model perturbations from a given nominal condition. Other control design specifications, including robustness, can be achieved via the concept of Closed-Loop Transfer Recovery [11]. This procedure, to be discussed further in Chapter 5, allows designers to achieve design performance and robustness starting from a satisfactory state-feedback

control law.

Development of reliable numerical optimization for linear optimal control synthesis requires effective algorithms for the evaluation of the objective function and the associated gradients. One numerical limitation in the design optimization scheme developed in [18] arises when degenerate modes occur in the closed-loop system. The corresponding system dynamics matrix will contain defective eigenvalues which are most often visualized in Jordan blocks. Occurrence of these defective eigenvalues becomes more frequent when a multiloop control law is being synthesized based on frequency-shaped design objectives or disturbances with degenerate power spectra (e.g. transverse Dryden turbulence spectra in wind turbulence). Degenerate systems can also occur in the design procedure of Closed-Loop Transfer Recovery (CLTR). Here, closed-loop dynamics from an output-feedback design in the CLTR procedure tend to overlap those attained under state feedback. With inexact arithmetic and large system matrices, these conditions lead to near degenerate systems and the appearance of Jordan blocks.

## 1.2 Key Research Motivation

- Experience with the optimization of low-order output feedback controllers in  $H^2$  inspires a search for a systematic way to include robustness in the design without tedious adjustment of design weights. One approach would be to use a different cost function, such as one based on worst-case initial conditions. Experience with this worst-case cost function indicates that while it would offer some help with robustness in particular, one would hope for a more comprehensive design methodology to obtain a more complete set of desired properties. Such is the promise of closed-loop transfer recovery (CLTR), which is based on recovering the properties of a state-feedback design with an output-feedback controller.
- Numerical optimization based on gradient information promises fast convergence, but it requires a precise calculation of the cost function and its gradient with respect to the controller design parameters. Existing numerical algorithms for these calculations lose accuracy as degeneracy in a system mode is approached and several of the corresponding eigenvalues become more defective. This loss of accuracy occurs in many design situations related to control of flexible structures with closely packed modes, or use of model matching and/or frequency-shaped objectives. Experience with closed-loop transfer recovery involving systems of reasonable size almost always generates defective eigenvalues.
- The terminology “more defective” is not common usage. However, this terminology does properly express the quantitative nature of the loss of accuracy that prevails in existing eigenvalue-eigenvector decomposition routines. One can define a tolerance, based on the condition of the eigenvector matrix, where inaccuracies in the established gradient calculation will noticeably affect the optimization process.
- In this research, several algorithms tolerant of defective systems have been considered. The most successful one involved converting the gradient computation to an exponential calculation and using the well-known Padé series for the matrix exponential. The Padé series itself is applied to a scaled matrix, and the careful term-by-term evaluation combined with a judicious scaling process have enabled this approach to handle a

wide range of system matrices. In addition to the theoretical assurances, this formula was demonstrated to work in many selected test cases.

- The proposed robust algorithm will provide an enabling technology for closed-loop transfer recovery within the framework of numerical optimization. System matrices under the CLTR design procedure tend to contain Jordan blocks owing to their size and the presence of overlapping poles. Within the CLTR framework, the best possible performance of an output-feedback controller can be predicted by examining the system theoretic properties with the Special Coordinate Basis (SCB). In addition, a low-order output-feedback controller can be employed to recover many state-feedback properties such as robustness without the tedious effort of manipulating penalty weights and the many ensuing design iterations. A demonstration of CLTR on a rotorcraft problem containing a reasonably large system model is presented. A systematic design procedure produces encouraging results over two flight conditions. The numerical optimization part of this procedure would not have been possible without the robust gradient formulation.
- A more ideal approach for gradient calculation aims at combining robustness of the new gradient algorithm with the speed, low memory usage, and scale insensitivity of the original diagonalization method [18]. With this comes the need of an alternate means of decomposing a matrix into its eigenvalues and eigenvectors—with the priority of keeping the condition of its eigenvector matrix above a predetermined level for computational reliability. Such an algorithm is sketched in Appendices A and B.

### 1.3 Summary of Contributions

The basic contribution is a robust algorithm for computing the integrals

$$\mathcal{X}(t) = \int_0^t e^{As} B e^{Cs} ds, \quad \mathcal{M}(t) = \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv,$$

which are at the heart of the gradient calculations for several types of cost functions. There are two main developments: first, the ability to express each integral as a matrix exponential, allowing well-known techniques for computing a matrix exponential to come into play; second, the rearrangement of the matrix exponential computations into a faster, less memory intensive, and more robust form specific for each integral.

Although the robust algorithm guarantees an answer for a wider range of input matrices, it uses more memory and is slower. The mode degeneracy is already parameterized in the condition of the eigenvector matrix. In running a wide variety of test cases, a limit on where this condition affects the optimization was determined. Using this limit, a means of switching between the original calculation and the robust one is made to promote optimum speed.

Another contribution is in implementing an approximate minimax cost function: unlike  $H^2$ -optimal control, this formulation minimizes a worst-case response to initial conditions. The actual worst case is approximated, and result of this approximation is in a form similar to that already used in  $H^2$ -norm.

Application of numerical optimization to CLTR completes a new development in CLTR design and analysis. A high-order and realistic design problem for the control of a UH-60 rotorcraft poses a challenging limit on the number of optimization trials one may pursue due

to the CPU requirements. It is shown that, when properly set up, CLTR using numerical optimization would allow designers to obtain satisfactory design results quickly and in a routine manner for output-feedback controllers starting from a satisfactory state-feedback design. While the required controller order may be higher in the CLTR design approach than in a pure direct optimization of a  $H^2$ -norm objective using an arbitrarily selected low-order controller structure, the need to re-design for robustness or other requirements is reduced.

## Chapter 2

# Control Design Using Numerical Optimization

### 2.1 Introduction

Direct parameter optimization provides a versatile method for linear multivariable control and has a broad range of applications. The design optimization is usually formulated within the context of  $H^2$  optimal control. For completeness, a general formulation of the control design problem is given in this chapter. Different design cost functions are defined corresponding to a class of deterministic and stochastic control problems. Equivalent relations between each design setup are established. Solutions of these design problems are a strong function of the disturbance input characteristics or plant initial conditions. These influences become particularly significant in the control synthesis of low-order controllers using direct parameter optimization. Such a design issue was addressed in [19] concerning the potential design sensitivity to plant initial conditions in optimal control synthesis. A worst-case design approach based on the largest singular value of a weighted covariance matrix was henceforth proposed by Bryson [19] to desensitize the optimal design. A simpler approach based on an upper bound to the worst-case cost is developed in this chapter and provides a convenient and numerically efficient way to address the worst-case design problem. A simple helicopter control design is used to illustrate the differences between a standard LQG design, an  $H^2$ -optimal control using parameter optimization, and those obtained using different design techniques for worst-case initial conditions.

### 2.2 Synthesis Model Description

The following problem formulation is suited for the control synthesis of a robust low-order controller in linear time-invariant systems. The system  $P^i(s)$  is to be controlled by a constant-gain controller  $C(s)$  as depicted in Figure 2.1 where  $z^i(s)$  is the controlled output vector,  $y_s^i(s)$  the measurement output vector,  $w^i(s)$  the disturbance input vector and  $u^i(s)$  the control input vector. For a consistent notation, the superscript  $i$  is used throughout to denote the system model at the  $i^{th}$  plant condition. Although there may be a set of plant models corresponding to a series of design conditions, there is only *one* controller,  $C(s)$ , in the design optimization. Hence, this single controller will provide stability and performance over *all* the defined plant conditions. The controller is modelled as a linear time-invariant system of a given pre-specified order. Its formulation can accomodate both a feedforward

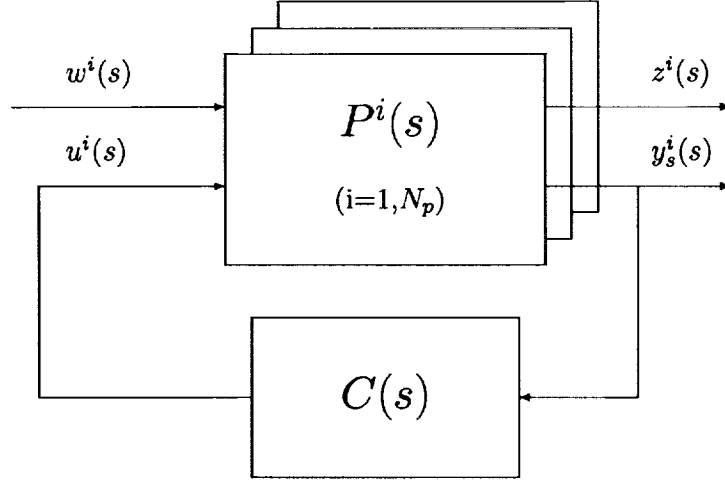


Figure 2.1: Closed-Loop System with a Feedback/Feedforward Controller

and a feedback controller structure. As stated before, plant parameter design robustness requirements in the context of our problem formulation are defined under the conditions that the control system  $C(s)$  stabilizes the class of plants  $P^i(s)$ ,  $1 \leq i \leq N_p$ .

State equations describing the system model  $P^i(s)$  at the  $i^{th}$  plant condition (Figure 2.1) are given in equations (2.1)-(2.3) below.

- State Equations:

$$\begin{cases} \dot{x}^i(t) = F^i x^i(t) + G^i u^i(t) + \Gamma^i w^i(t) \\ x^i(0) = 0 \end{cases} \quad (2.1)$$

where  $x^i(t)$  is a  $n \times 1$  plant state vector,  $u^i(t)$  an  $m \times 1$  control vector,  $w^i(t)$  an  $m' \times 1$  disturbance-input vector,  $F^i$  an  $n \times n$  state matrix,  $G^i$  an  $n \times m$  control distribution matrix and  $\Gamma^i$  an  $n \times m'$  input-disturbance distribution matrix.

Notice that, in the above description, we assume that all the system states are initially acquiescent. This assumption is made without loss of generality since one can always establish impulsive inputs  $w^i(t)$  together with an appropriate influence matrix  $\Gamma^i$  to generate any given state initial conditions.

- Measurement Equations:

$$y_s^i(t) = H_s^i x^i(t) + D_{su}^i u^i(t) + D_{sw}^i w^i(t) \quad (2.2)$$

where  $y_s^i(t)$  is a  $p \times 1$  measurement vector,  $H_s^i$  a  $p \times n$  state-to-measurement distribution matrix,  $D_{su}^i$  a  $p \times m$  control-to-measurement distribution matrix and  $D_{sw}^i$  a  $p \times m'$  input-disturbance-to-measurement distribution matrix.

- Criterion Equations:

$$z^i(t) = H_c^i x^i(t) + D_{cu}^i u^i(t) + D_{cw}^i w^i(t) \quad (2.3)$$

where  $z^i(t)$  is a  $p' \times 1$  criterion vector,  $H_c^i$  a  $p' \times n$  state-to-criterion distribution matrix,  $D_{cu}^i$  a  $p' \times m$  control-to-criterion distribution matrix and  $D_{cw}^i$  a  $p' \times m'$  input-disturbance-to-criterion distribution matrix.

For generality, the disturbances  $w^i(t)$  are modeled as outputs of a linear time-invariant system excited by either impulse inputs or white noise signals. In this manner, one can shape the disturbance signals to have any deterministic response (e.g. filtered step functions, sinusoidal functions, exponentially decayed or growing sinusoidal functions, etc...) or, to model stochastic inputs with any given power spectral density functions. At the  $i^{th}$  plant condition, the disturbance model is given by equations (2.4)-(2.5) below.

- Disturbance State Equations:

$$\begin{cases} \dot{x}_w^i(t) = F_w^i x_w^i(t) + \Gamma_w^i \eta^i(t) \\ x_w^i(0) = 0 \end{cases} \quad (2.4)$$

where  $x_w^i(t)$  is a  $n' \times 1$  disturbance state vector,  $\eta^i(t)$  a  $m' \times 1$  vector of either impulses (i.e.,  $\eta^i(t) = \eta_o^i \delta(t)$  with  $E[\eta_o^i] = 0$  and  $E[\eta_o^i \eta_o^{iT}] = W_o^i$ ), or white-noise processes  $\eta(t)$  with zero mean and covariance  $E[\eta_o^i(t) \eta_o^{iT}(\tau)] = W_o^i \delta(t - \tau)$ . The covariance matrix  $W_o^i$  is an  $m' \times m'$  diagonal positive semi-definite matrix,  $F_w^i$  an  $n' \times n'$  state matrix of the disturbance model and  $\Gamma_w^i$  an  $n' \times m'$  input-distribution matrix.

- Disturbance Output Equations:

$$w^i(t) = H_w^i x_w^i(t) + D_w^i \eta^i(t) \quad (2.5)$$

where  $w^i(t)$  is a  $m' \times 1$  disturbance output vector,  $H_w^i$  an  $m' \times n'$  disturbance output matrix and  $D_w^i$  an  $m' \times m'$  direct feedthrough distribution matrix.

State model of the controller  $C(s)$  (Figure 2.1) is that of a linear time-invariant system described by equations (2.6)-(2.7) below.

- Controller State Equations:

$$\begin{cases} \dot{x}_c(t) = A_c x_c(t) + B_c y_s^i(t) \\ x_c(0) = 0 \end{cases} \quad (2.6)$$

where  $x_c(t)$  is a  $r \times 1$  controller state vector,  $A_c$  a  $r \times r$  state matrix of the controller and  $B_c$  a  $r \times p$  measurement-input distribution matrix.

- Control Equations:

$$u^i(t) = C_c x_c(t) + D_c y_s^i(t) \quad (2.7)$$

where  $u^i(t)$  is an  $m \times 1$  feedback control vector,  $C_c$  an  $m \times r$  control-output distribution matrix and  $D_c$  an  $m \times p$  direct feedthrough matrix.

For control-law synthesis, any elements of the controller state matrices can be chosen as design parameters while the remaining elements can be left fixed at pre-assigned values. In addition, if needed, linear and nonlinear equality or inequality constraints can be established among the selected design parameters in order to ensure additional design structure. For convenience in the derivation of the performance index and its gradients with respect to the controller design parameters, we define a matrix  $C_o$  that assembles all the controller state matrices ( $A_c, B_c, C_c, D_c$ ) into one compact form as follows,

$$C_o = \begin{bmatrix} D_c & C_c \\ B_c & A_c \end{bmatrix}_{(m+r) \times (p+r)} \quad (2.8)$$

Thus, the single matrix  $C_o$  will completely define the controller state model. Obviously, for the case of a static output-feedback design (i.e., the controller order  $r = 0$ ), we simply have  $C_o = D_c$ . In the following sections, we will develop a control design problem based on the minimization of a performance objective using the controller  $C(s)$  defined in equations (2.6)-(2.7).

## 2.3 Formulation of the Closed-Loop System

In the general case, the problem formulation assumes that there is no implicit-loop paths within the feedback control system. Namely, the control input  $u^i(t)$  or the measurement output  $y_s^i(t)$  must not have any direct link to itself. This translates into the conditions that one of the products  $D_c D_{su}^i$  or  $D_{su}^i D_c$  must be zero. These conditions are not restrictive since, in practice, the presence of either actuation or sensor dynamics would automatically result in a system that satisfies the above assumptions. Moreover, for *well-posed* systems one can always reformulate the problem into an equivalent problem involving a modified set of measurement outputs  $\bar{y}_s^i(t) (= y_s^i(t) - D_{su}^i u(t))$ . Thus,

$$\bar{y}_s^i(t) = H_s^i x^i(t) + D_{sw}^i w^i(t) \quad (2.9)$$

Note that the re-defined measurement outputs  $\bar{y}_s^i(t)$  do not have a term involving the control variable  $u^i(t)$  (i.e.,  $D_{su}^i = 0$ ). However, for the discussions that follow, we make use of the results in [18] where we assume that either  $D_{su}^i D_c = 0$  or  $D_c D_{su}^i = 0$ .

### 2.3.1 Case $D_c D_{su}^i = 0$

Combining the states of the plant, controller, and disturbances into an augmented system with the following states

$$x^i(t) = \begin{bmatrix} x^i(t) \\ x_c^i(t) \\ x_w^i(t) \end{bmatrix}. \quad (2.10)$$

Dynamics for the overall closed-loop system are

$$\dot{x}^i(t) = F^i x^i(t) + \Gamma^i \eta^i(t), \quad (2.11)$$

where

$$F^i = \begin{bmatrix} F^i + G^i D_c H_s^i & G^i C_c & (\Gamma^i + G^i D_c D_{sw}^i) H_w^i \\ B_c(I + D_{su}^i D_c) H_s^i & A_c + B_c D_{su}^i C_c & B_c(I + D_{su}^i D_c) D_{sw}^i H_w^i \\ 0 & 0 & F_w^i \end{bmatrix}_{(n+r+n') \times (n+r+n')} \quad (2.12)$$

$$\Gamma^i = \begin{bmatrix} (\Gamma^i + G^i D_c D_{sw}^i) D_w^i \\ B_c(I + D_{su}^i D_c) D_{sw}^i D_w^i \\ \Gamma_w^i \end{bmatrix}_{(n+r+n') \times m'} \quad (2.13)$$

$$H_s^i = \begin{bmatrix} (I + D_{su}^i D_c) H_s^i & D_{su}^i C_c & (I + D_{su}^i D_c) D_{sw}^i H_w^i \end{bmatrix}_{p \times (n+r+n')} \quad (2.14)$$

$$D_s^i = \begin{bmatrix} (I + D_{su}^i D_c) D_{sw}^i D_w^i \end{bmatrix}_{p \times m'} \quad (2.15)$$

$$H_c^i = \begin{bmatrix} H_c^i + D_{cu}^i D_c H_s^i & D_{cu}^i C_c & (D_{cu}^i D_c D_{sw}^i + D_{cw}^i) H_w^i \end{bmatrix}_{p' \times (n+r+n')} \quad (2.16)$$

and

$$C^i = \begin{bmatrix} D_c H_s^i & C_c & D_c D_{sw}^i H_w^i \end{bmatrix}_{m \times (n+r+n')} \quad (2.17)$$

With the above definitions, equations (2.2), (2.3), and (2.7) for  $y_s^i(t)$ ,  $z^i(t)$  and  $u^i(t)$  become

$$y_s^i(t) = H_s^i x^i(t) + D_{sw}^i w^i(t) \quad (2.18)$$

$$z^i(t) = H_c^i x^i(t) + (D_{cu}^i D_c D_{sw}^i + D_{cw}^i) D_w^i \eta^i(t) \quad (2.19)$$

$$u^i(t) = C^i x^i(t) + D_c D_{sw}^i D_w^i \eta^i(t) \quad (2.20)$$

### 2.3.2 Case $D_{su}^i D_c = 0$

The closed-loop system for the combined plant, controller, and disturbance dynamics is again defined by

$$\dot{x}^i(t) = F^i x^i(t) + \Gamma^i \eta^i(t), \quad (2.21)$$

where the system matrix  $F^i$  and the distribution matrix  $\Gamma^i$  are now given by

$$F^i = \begin{bmatrix} F^i + G^i D_c H_s^i & G^i (I + D_c D_{su}^i) C_c & (\Gamma^i + G^i D_c D_{sw}^i) H_w^i \\ B_c H_s^i & A_c + B_c D_{su}^i C_c & B_c D_{sw}^i H_w^i \\ 0 & 0 & F_w^i \end{bmatrix} \quad (2.22)$$

$$\Gamma^i = \begin{bmatrix} (\Gamma^i + G^i D_c D_{sw}^i) D_w^i \\ B_c D_{sw}^i D_w^i \\ \Gamma_w^i \end{bmatrix}. \quad (2.23)$$

Equations (2.18), (2.19), and (2.20) for the sensor, control, and criterion outputs are defined as before, where the matrices  $H_s^i$ ,  $D_s^i$ ,  $H_c^i$ , and  $C^i$  take on the following form

$$H_s^i = \begin{bmatrix} H_s^i & D_{su}^i C_c & D_{sw}^i H_w^i \end{bmatrix} \quad (2.24)$$

$$D_s^i = \begin{bmatrix} D_{sw}^i D_w^i \end{bmatrix} \quad (2.25)$$

$$H_c^i = [H_c + D_{cu} D_c H_s \quad D_{cu} (I + D_c D_{su}) C_c \quad (D_{cu} D_c D_{sw} + D_{cw}) H_w], \quad (2.26)$$

and

$$C^i = \begin{bmatrix} D_c H_s^i & (I + D_c D_{su}^i) C_c & D_c D_{sw}^i H_w^i \end{bmatrix}. \quad (2.27)$$

## 2.4 Design Cost Functions for $H^2$ Optimal Control

To examine the entire class of  $H^2$ -optimal control problems and to handle the problem of sensitivity to plant modeling uncertainties, we define the objective function given in equations (2.28) and (2.32). This formulation turns out to be versatile and well-posed for the setting of a nonlinear constrained optimization problem. However, depending on the types of disturbance model, that is whether the disturbance outputs  $w^i(t)$  are responses to impulse or white-noise inputs, different definitions of the objective function are needed. Another cost function is defined in equation (2.38) to address a worst-case control design problem.

For these cost functions to be well-defined (bounded), one needs to identify the presence of direct feedthrough terms from the disturbance inputs  $\eta^i(t)$  to the criterion outputs  $z^i(t)$ . Specifically, for either impulsive inputs or white-noise disturbances, the objective functions defined in equations (2.28), (2.32), and (2.38) would become unbounded when the direct feedthrough term  $D_{cw}^i$  is nonzero. When a direct feedthrough term exists (e.g., in command-following synthesis), the disturbance inputs must be implemented as band-limited signals (i.e., they are generated from outputs of some roll-off shaping filters). Basically, this reduces the direct term  $D_{cw}^i$  to zero and places the direct influence of the disturbances within the criterion output distribution matrix  $H_c^i$  in equation (2.16).

A similar problem would occur for the case where there is a direct feedthrough term between the disturbances  $w(t)$  and the measurements  $y_s(t)$ . The condition for a bounded performance objective requires that the products  $Q^i (D_{cu}^i D_c D_{sw}^i + D_{cw}^i)$  and  $R^i D_c D_{sw}^i$  be

zero for the cases of either impulsive or white-noise disturbances. Again, such conditions will always hold for band-limited sensor noises.

These cost functions are given using the terminology defined in the previous section for the closed-loop systems.

### 2.4.1 Random Impulsive Disturbances

Consider the following cost function

$$J_1(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \int_0^{t_f} e^{2\alpha^i t} E[z^{iT}(t) Q^i z^i(t) + u^{iT}(t) R^i u^i(t)] dt \quad (2.28)$$

The expectation operator  $E[-]$  is over the ensemble of the random variables  $\eta_o^i$  in the parameterized impulse inputs  $\eta^i(t) = \eta_o^i \delta(t)$ . Control design problems formulated with the above performance index  $J(t_f, C_o)$  are often classified under the category of deterministic control. Under this category are, for example, the familiar control problems of command tracking control, disturbance rejection of unwanted but known external input signals, implicit and explicit model-following designs,  $H^2$ -control to initial conditions and  $H^\infty$ -control to sinusoidal inputs.

This form for the cost function can be further expanded by embedding the closed-loop system responses

$$x^i(t) = e^{(F^i + \alpha^i I)t} \Gamma^i \eta_o^i \quad (2.29)$$

directly into the cost function  $J_1(t_f, C_o)$  as

$$\begin{aligned} J_1(t_f, C_o) &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \int_0^{t_f} E \left[ \eta_o^{iT} \Gamma^{iT} e^{(F^i + \alpha^i I)^T t} [H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i] e^{(F^i + \alpha^i I)^T t} dt \Gamma^i \eta_o^i \right] \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i E \left[ \eta_o^{iT} \Gamma^{iT} S^i(t_f, C_o) \Gamma^i \eta_o^i \right] \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \Gamma^{iT} S^i(t_f, C_o) \Gamma^i E \left[ \eta_o^i \eta_o^{iT} \right] \right\} \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \Gamma^{iT} S^i(t_f, C_o) \Gamma^i E \left[ \eta_o^i \eta_o^{iT} \right] \right\} \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i \right\} \end{aligned} \quad (2.30)$$

where

$$S^i(t_f, C_o) = \int_0^{t_f} e^{(F^i + \alpha^i I)^T t} [H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i] e^{(F^i + \alpha^i I)^T t} dt \quad (2.31)$$

and  $W_o^i = E \left[ \eta_o^i \eta_o^{iT} \right]$ .

### 2.4.2 Random White-Noise Disturbances

The objective function defined in equation (2.28) is no longer valid since it is unbounded due to the presence of disturbances with white-noise spectra. For this reason, it is not proper to mix, in a given single plant condition, design considerations of initial conditions

and of disturbance rejection to white-noise disturbances with the same objective function. Alternate objective functions for the disturbance rejection problem must be defined (Refer to Appendix A of Ref. [18]). There are basically two ways to define a quadratic objective function for white-noise disturbances. The first formulation involves the state covariance responses evaluated at the terminal time  $t_f$ . Namely,

$$J_2(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i E_{\alpha^i} [z^{iT}(t_f) Q^i z^i(t_f) + u^{iT}(t_f) R^i u^i(t_f)]. \quad (2.32)$$

The second formulation is a time-average of the state covariance responses,

$$J_3(t_f, C_o) = \frac{1}{2t_f} \sum_{i=1}^{N_p} W_p^i E_{\alpha^i} \left[ \int_0^{t_f} z^{iT}(t) Q^i z^i(t) + u^{iT}(t) R^i u^i(t) dt \right]. \quad (2.33)$$

The expectation operator  $E_{\alpha^i}[-]$  is over the ensemble of the random processes defined in the input variables  $\eta^i(t)$  for a closed-loop system destabilized by a factor  $\alpha^i$ . The destabilization effectively adds a value  $\alpha^i$  to the diagonal elements of the closed-loop system matrix. Given one of the above performance indices, one can address the entire class of  $H^2$ -norm based control design problems. For example, we can solve for the linear quadratic regulator design (LQR), the linear quadratic gaussian (LQG) design, loop transfer recovery (LTR), closed-loop transfer recovery (CLTR), or model reduction based on the minimization of the  $H^2$ -norm of the error.

These two cost functions can be shown to be related to each other and also to the cost  $J_1(t_f, C_o)$  defined in Section 2.4.1. Embedding the closed-loop state responses,

$$x^i(t) = \int_0^{t_f} e^{(F^i + \alpha^i I)(t-\tau)} \Gamma^i \eta^i(\tau) d\tau, \quad (2.34)$$

into  $J_2(t_f, C_o)$ , we have

$$J_2(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \int_0^{t_f} \int_0^{t_f} e^{(F^i + \alpha^i I)(t_f-\tau)} \Gamma^i E \left[ \eta^i(\tau) \eta^{iT}(s) \right] \Gamma^{iT} e^{(F^i + \alpha^i I)^T(t_f-s)} d\tau ds \right\} \quad (2.35)$$

With  $E \left[ \eta^i(\tau) \eta^{iT}(s) \right] = W_o^i \delta(\tau - s)$ , equation (2.35) becomes

$$\begin{aligned} J_2(t_f, D_c) &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \int_0^{t_f} e^{(F^i + \alpha^i I)(t_f-\tau)} \Gamma^i W_o^i \Gamma^{iT} e^{(F^i + \alpha^i I)^T(t_f-\tau)} d\tau \right\} \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \int_0^{t_f} e^{(F^i + \alpha^i I)t} \Gamma^i W_o^i \Gamma^{iT} e^{(F^i + \alpha^i I)^T t} dt \right\} \\ &= \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \Gamma^{iT} \int_0^{t_f} e^{(F^i + \alpha^i I)^T t} \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) e^{(F^i + \alpha^i I)t} dt \Gamma^i W_o^i \right\} \end{aligned}$$

Or simplifying,

$$J_2(t_f, D_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \operatorname{tr} \left\{ \Gamma^{iT} S^i(t_f, D_o) \Gamma^i W_o^i \right\}. \quad (2.36)$$

The above objective function is identical to the cost function  $J_1(t_f, D_o)$  in equation (2.30).

As before, the objective function  $J_3(t_f, C_o)$  can be simplified using the closed-loop responses given in equation (2.34) as follows,

$$J_3(t_f, C_o) = \frac{1}{2t_f} \sum_{i=1}^{N_p} W_p^i \operatorname{tr} \left\{ \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \Phi(t_f) \right\} \quad (2.37)$$

where

$$\Phi(t_f) = \int_0^{t_f} \int_0^t \int_0^t e^{(F^i + \alpha^i I)(t-\tau)} \Gamma^i E \left[ \eta^i(\tau) \eta^{iT}(s) \right] \Gamma^{iT} e^{(F^i + \alpha^i I)^T(t-s)} d\tau ds dt$$

Again, with  $E[w(\tau)w^T(s)] = W_o \delta(\tau - s)$ ,

$$\begin{aligned} J_3(t_f, C_o) &= \frac{1}{2t_f} \sum_{i=1}^{N_p} W_p^i \operatorname{tr} \left\{ \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \right. \\ &\quad \left. \int_0^{t_f} \int_0^t e^{(F^i + \alpha^i I)(t-\tau)} \Gamma^{iT} W_o^i \Gamma^i e^{(F^i + \alpha^i I)^T(t-\tau)} d\tau dt \right\} \\ &= \frac{1}{2t_f} \sum_{i=1}^{N_p} W_p^i \operatorname{tr} \left\{ \Gamma^{iT} \int_0^{t_f} \int_0^t e^{(F^i + \alpha^i I)^T(t-\tau)} \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) \right. \\ &\quad \left. e^{(F^i + \alpha^i I)(t-\tau)} d\tau dt \Gamma^i W_o^i \right\} \\ &= \frac{1}{t_f} \int_0^{t_f} J_2(t, C_o) dt \leq J_2(t_f, C_o) \end{aligned}$$

since the objective function  $J_2(t, C_o)$  is a monotonic increasing function of time  $t$ . Hence,  $J_3(t_f, C_o) \leq J_2(t_f, C_o)$  for any finite terminal time  $t_f$ , and they become equal in the steady-state case where  $t_f \rightarrow \infty$ .

We have shown that a design objective for random white-noise disturbances as given in the cost functional  $J_2(t_f, C_o)$  or  $J_3(t_f, C_o)$  is equivalent to one involving initial conditions or impulsive disturbances.

### 2.4.3 Worst-Case Impulsive Disturbances

A worst-case design objective to impulsive disturbances, i.e.  $\eta^i(t) = \eta_o^i \delta(t)$  is defined as follows,

$$J_\sigma(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \max_{\eta_o^i} \frac{\int_0^{t_f} [z^{iT}(t) Q^i z^i(t) + u^{iT}(t) R^i u^i(t)] dt}{\eta_o^{iT} \eta_o^i} \quad (2.38)$$

Substituting the closed-loop system responses given in equation (2.29), we have

$$J_\sigma(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \max_{\eta_o^i} \frac{\eta_o^{iT} \Gamma^{iT} S^i(t_f, C_o) \Gamma^i \eta_o^i}{\eta_o^{iT} \eta_o^i}.$$

The control problem for worst-case impulsive disturbances is to minimize the maximum eigenvalue of  $\Gamma^{iT} S^i(t_f, C_o) \Gamma^i$ , or equivalently its maximum singular value since the matrix  $\Gamma^{iT} S^i(t_f, C_o) \Gamma^i$  is symmetric and positive semi-definite. It is well-known that when singular values of a matrix are repeated, they are not differentiable with respect to elements of the matrix ([23], p.288). With the relation  $\bar{\sigma}^2(A) \leq \text{tr}\{A^T A\}$  for any arbitrary matrix  $A$ , we can re-define the worst-case functional by its upper bound,

$$\begin{aligned} J_{\bar{\sigma}}(t_f, C_o) \leq \bar{J}_{\bar{\sigma}}(t_f, C_o) &= \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( \Gamma^{iT} S^i(t_f, C_o) \Gamma^i \right)^2 \right\} \\ &= \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( S^i(t_f, C_o) \Gamma^i \Gamma^{iT} \right)^2 \right\} \end{aligned}$$

where  $S^i(t_f, C_o)$  is defined in equation (2.31). Derivatives of this upper bound are easy to compute. Thus, we prefer this approximation based on the Frobenius norm to the exact worst-case design objective. While it is usually best to scale the disturbances through the  $\Gamma^i$  matrix, in practice, it is often convenient to allow an additional weighting matrix  $W_o$  in the matrix  $\Gamma^i \Gamma^{iT}$  as  $\Gamma^i W_o^i \Gamma^{iT}$ . Equation (2.39) becomes

$$\bar{J}_{\bar{\sigma}}(t_f, C_o) = \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} \right)^2 \right\} \quad (2.39)$$

This corresponds to a bound on the maximum singular value problem

$$J_{\bar{\sigma}}(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \max_{\eta_o^i} \frac{\eta_o^{iT} W_o^{i\frac{1}{2}} \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^{i\frac{1}{2}} \eta_o^i}{\eta_o^{iT} \eta_o^i}.$$

The problem formulation for worst-case disturbances can also be extended to address cases where the disturbances have a given deterministic time behaviour. In this situation, the control problem is then to find an output-feedback controller for these specific types of disturbances (e.g., steps, ramps, sinusoidal inputs of known frequencies, and others).

#### 2.4.4 Worst-Case for White-Noise Disturbances

The usual convention for white-noise disturbances is that the components to the disturbance vector  $\eta^i(t)$  are independent, and since  $E[\eta^i(\tau) \eta^{iT}(s)] = W_o^i \delta(\tau - s)$ , this implies that  $W_o^i$  is full rank. However, for the worst-case design we optimize over  $W_o^i$  that are rank 1, thus  $W_o^i = \eta_o^i \eta_o^{iT}$  corresponds to a disturbance of the form  $\eta^i(t) = \eta_o^i \rho^i(t)$ , where  $\rho^i(t)$  is a scalar white noise source of unit spectral density. Within this context we can write the two forms of the worst-case cost functional for white noise (analogous to the two forms of the  $H^2$  cost for white noise). The first is related to the state covariance response evaluated at the time  $t_f$

$$J_2^{wc}(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \max_{W_o^i} \frac{E_{\alpha^i} [z^{iT}(t_f) Q^i z^i(t_f) + u^{iT}(t_f) R^i u^i(t_f)]}{\text{tr} \{W_o^i\}} \quad (2.40)$$

The second is an average of the worst-case state covariance response in the time interval  $[0, t_f]$

$$J_3^{wc}(t_f, C_o) = \frac{1}{2t_f} \sum_{i=1}^{N_p} W_p^i \max_{W_o^i} \frac{E_{\alpha^i} \int_0^{t_f} [z^{iT}(t) Q^i z^i(t) + u^{iT}(t) R^i u^i(t)] dt}{\text{tr} \{W_o^i\}} \quad (2.41)$$

By embedding the dynamics, both cost functionals can be written in the form

$$J_{\bar{\sigma}}(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \max_{W_o^i} \frac{\text{tr} \{ \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i \}}{\text{tr} \{ W_o^i \}}$$

The cost driven by such a noise corresponds to the maximum singular value of  $\Gamma^{iT} S^i(t_f, C_o) \Gamma^i$  and would be an upper bound to the regular  $H^2$  cost defined for white noise.

As before, we use the Frobenius norm of  $\Gamma^{iT} S^i(t_f, C_o) \Gamma^i$  to bound its maximum singular value. The result is the same as given in equation (2.39).

Control-law synthesis using the above worst-case objectives and properties of their corresponding solutions are not well understood and need further investigation. This is a subject left for future research. However, it seems to have a potential of providing controller designs that are insensitive to plant disturbances in applications such as model reduction and closed-loop transfer recovery.

#### 2.4.5 Design Features of Direct Optimization

Note that the performance indices given in equations (2.28), (2.32), and (2.38) are evaluated to a finite-time horizon  $t_f$ . The use of a finite time plays a significant role in the implementation of a reliable design algorithm for the optimum steady-state solution. It should be recognized that the objective function is well-defined regardless of whether the feedback control-law is stabilizing or not. Furthermore, a class of problems associated with command tracking of neutrally stable or unstable target responses (e.g. step and ramp commands, sinusoidal trajectories) is only tractable under the setting of a finite-time objective function, but not in the confine of a steady-state objective function where  $t_f \rightarrow \infty$ . In practice, whenever possible, steady-state results are usually achieved when the terminal time  $t_f$  is equal to five or six times the slowest time constant in the closed-loop system responses.

Besides the concept of design based on a finite terminal time  $t_f$ , other unique and important features have also been incorporated into the design objective function of equations (2.28), (2.32), and (2.38). First of all, this objective function is not the usual quadratic cost function traditionally defined in linear optimal control. It is instead a weighted average of quadratic performance indices evaluated over a set of design conditions ( $1 \leq i \leq N_p$ ). Different weights are assigned to each plant condition through the scalar variable  $W_p^i$  where  $W_p^i \geq 0$ . Of course, design for only one plant condition (with  $N_p = 1$ ) reduces to the usual quadratic cost function evaluated at the *nominal* design condition. The time-weighted factor  $e^{2\alpha^i t}$  allows us to impose directly a stability constraint on the closed-loop eigenvalues in the  $H^2$ -optimal control problems. Namely, when a steady-state design has been achieved and the optimum objective function is bounded, then the closed-loop system eigenvalues for the controllable modes will have real parts less than  $-\alpha^i$ . Finally, the weighting matrices  $Q^i$  and  $R^i$  are symmetric and positive *semi-definite* matrices. Note that our solution approach to the minimization of the objective function  $J(t_f, C_o)$  is based on nonlinear optimization; hence, it does not require the control weighting matrix  $R^i$  to be positive definite. In fact, in some design problems such as command tracking and model reduction, a proper objective function contains only penalties on the tracking or model-matching errors and does not include penalties on the control variables (i.e., with  $R^i = 0$ ). The above observation further indicates the applicability of the design procedure in addressing the significant class of solvable *singular* optimal control problems [33].

## 2.5 Derivation of Cost Function Gradients

In this section, explicit gradient expressions for the cost functions defined in Section 2.4 with respect to the controller matrix  $C_o$  are derived. Results are given for both cases  $D_c D_{su} = 0$  and  $D_{su} D_c = 0$ . To begin, we consider the following useful differentiation rules of the trace function,

$$\frac{\partial}{\partial C_o} \text{tr} \{ Q C_o \mathcal{R} \} = (\mathcal{R} Q)^T \quad \text{and} \quad \frac{\partial}{\partial C_o} \text{tr} \{ Q C_o^T \mathcal{R} \} = \mathcal{R} Q$$

In addition, differentiation of the exponential function with respect to its argument can be obtained as follows. For a linear time-invariant system,

$$\begin{aligned} \dot{\phi}(t) &= A\phi(t) \\ \phi(0) &= I \end{aligned}$$

Then,

$$\frac{\partial \phi}{\partial C_o}(t) = A \frac{\partial \phi}{\partial C_o}(t) + \frac{\partial A}{\partial C_o} \phi(t)$$

Solving for  $\partial \phi / \partial C_o$ , we obtain

$$\frac{\partial \phi}{\partial C_o}(t) = \int_0^t \phi(t - \sigma) \frac{\partial A}{\partial C_o} \phi(\sigma) d\sigma$$

With the following definitions (as suggested in [18]),

$$F_o^i = \begin{bmatrix} F^i & 0 & \Gamma^i H_w^i \\ 0 & 0 & 0 \\ 0 & 0 & F_w^i \end{bmatrix}_{(n+r+n') \times (n+r+n')} \quad (2.42)$$

$$G_o^i = \begin{bmatrix} G^i & 0 \\ 0 & I \\ 0 & 0 \end{bmatrix}_{(n+r+n') \times (m+r)} \quad (2.43)$$

$$\Gamma_o^i = \begin{bmatrix} \Gamma^i D_w^i \\ 0 \\ \Gamma_w^i \end{bmatrix}_{(n+r+n') \times m'} \quad (2.44)$$

$$H_o^i = \begin{bmatrix} H_s^i & 0 & D_{sw}^i H_w^i \\ 0 & I & 0 \end{bmatrix}_{(p+r) \times (n+r+n')} \quad (2.45)$$

$$H_1^i = \begin{bmatrix} H_c^i & 0 & D_{cw}^i H_w^i \end{bmatrix}_{p' \times (n+r+n')} \quad (2.46)$$

$$D_o^i = \begin{bmatrix} D_{sw}^i D_w^i \\ 0 \end{bmatrix}_{(p+r) \times m'} \quad (2.47)$$

$$D_1^i = \begin{bmatrix} D_{su}^i & 0 \\ 0 & 0 \end{bmatrix}_{(p+r) \times (m+r)} \quad (2.48)$$

$$D_2^i = \begin{bmatrix} D_{cu}^i & 0 \end{bmatrix}_{p' \times (m+r)} \quad (2.49)$$

$$T_1 = [I \ 0]_{m \times (m+r)} \quad (2.50)$$

$$T_2 = \begin{bmatrix} 0 & 0 \\ 0 & I \\ 0 & 0 \end{bmatrix}_{(n+r+n') \times (m+r)} \quad (2.51)$$

$$T_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & I & 0 \end{bmatrix}_{(p+r') \times (n+r+n')} \quad (2.52)$$

Gradients of the respective cost functions can then be expressed in a convenient form.

### 2.5.1 Gradients of $J_1(t_f, C_o)$ (Case $D_{su}D_c = 0$ )

Define

$$F'^i = F_o^i + (G_o^i + T_2 C_o D_1^i) C_o H_o^i \quad (2.53)$$

$$\Gamma'^i = \Gamma_o^i + (G_o^i + T_2 C_o D_1^i) C_o D_o^i \quad (2.54)$$

$$H_c^i = H_1^i + D_2^i C_o H_o^i \quad (2.55)$$

$$C^i = T_1 C_o H_o^i \quad (2.56)$$

then

$$\begin{aligned} \frac{\partial J_1}{\partial C_o}(t_f, C_o) = & \sum_{i=1}^{N_p} W_p^i (D_2^{iT} Q H_c^i + T_1^T R C^i) \mathcal{X}^i(t_f, C_o) H_o^{iT} \\ & + (G_o^i + T_2 C_o D_1^i)^T [\mathcal{M}^i(t_f, C_o) H_o^{iT} + S^i(t_f, C_o) \Gamma'^i W_o^i D_o^{iT}] \\ & + T_2^T [\mathcal{M}^i(t_f, C_o) H_o^{iT} + S^i(t_f, C_o) \Gamma'^i W_o^i D_o^{iT}] (D_1^i C_o)^T \end{aligned} \quad (2.57)$$

where

$$\mathcal{X}^i(t_f, C_o) = \int_0^{t_f} e^{(F'^i + \alpha^i I)t} \Gamma'^i W_o^i \Gamma'^{iT} e^{(F'^i + \alpha^i I)^T t} dt \quad (2.58)$$

$$S^i(t_f, C_o) = \int_0^{t_f} e^{(F'^i + \alpha^i I)^T t} (H_c^{iT} Q H_c^i + C^{iT} R C^i) e^{(F'^i + \alpha^i I)t} dt \quad (2.59)$$

$$\begin{aligned} \mathcal{M}^i(t_f, C_o) = & \int_0^{t_f} \int_0^t e^{(F'^i + \alpha^i I)^T (t-\sigma)} (H_c^{iT} Q H_c^i + C^{iT} R C^i) e^{(F'^i + \alpha^i I)t} \Gamma'^i W_o^i \Gamma'^{iT} \\ & e^{(F'^i + \alpha^i I)^T \sigma} d\sigma dt \end{aligned} \quad (2.60)$$

### 2.5.2 Gradients of $J_1(t_f, C_o)$ (Case $D_c D_{su} = 0$ )

Define

$$F'^i = F_o^i + G_o^i C_o (H_o^i + D_1^i C_o T_3) \quad (2.61)$$

$$\Gamma'^i = \Gamma_o^i + G_o^i C_o D_o^i \quad (2.62)$$

$$H_c^i = H_1^i + D_2^i C_o (H_o^i + D_1^i C_o T_3) \quad (2.63)$$

$$C^i = T_1 C_o (H_o^i + D_1^i C_o T_3) \quad (2.64)$$

then

$$\begin{aligned} \frac{\partial J_1}{\partial C_o}(t_f, C_o) = & \sum_{i=1}^{N_p} W_p^i [(D_2^{iT} Q H_c^i + T_1^T R C^i) \mathcal{X}^i(t_f, C_o) (H_o^i + D_1^i C_o T_3)^T \\ & + G_o^{iT} \mathcal{M}^i(t_f, C_o) [H_o^i + D_1^i C_o T_3]^T + (G_o^i C_o D_1^i)^T \mathcal{M}^i(t_f, C_o) T_3^T + G_o^{iT} S(t_f, C_o) \Gamma'^i W_o^i D_o^{iT} \end{aligned} \quad (2.65)$$

### 2.5.3 Gradients of $\tilde{J}_\sigma(t_f, C_o)$

The upper bound  $\tilde{J}_\sigma$  based on the Frobenius norm is a continuous and differentiable function of the controller design parameters. Computational effort in the gradients of  $\tilde{J}_\sigma(t_f, C_o)$  is essentially the same as that for the cost function  $J_1(t_f, C_o)$ . Recall from equation (2.39) that

$$\tilde{J}_\sigma(t_f, C_o) = \frac{1}{2} \sum_{i=1}^{N_p} W_p^i \text{tr} \left\{ \left( S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} \right)^2 \right\} \quad (2.66)$$

Next, we derive the gradients of  $\tilde{J}_\sigma(t_f, C_o)$  for the cases  $D_{su}D_c = 0$  and  $D_cD_{su} = 0$ .

#### 2.5.4 Case $D_{su}D_c = 0$

Using the definitions of  $F^i$ ,  $\Gamma^i$ ,  $H_c^i$ , and  $C^i$  given in equations (2.53)-(2.56), we have

$$\begin{aligned} \frac{\partial \tilde{J}_\sigma(t_f, C_o)}{\partial C_o} &= \sum_{i=1}^{N_p} W_p^i \left( D_2^{iT} Q^i H_c^i + T_1^T R^i C^i \right) \mathcal{X}_{WS}^i(t_f, C_o) H_o^{iT} \\ &\quad + (G_o^i + T_2 C_o D_1^i)^T \mathcal{M}_{WS}^i(t_f, C_o) H_o^{iT} + T_2^T \mathcal{M}_{WS}^i(t_f, C_o) (D_1^i C_o H_o^i)^T \\ &\quad + (G_o^i + T_2 C_o D_1^i)^T S(t_f) \Gamma^i W_o^i \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i D_o^{iT} \\ &\quad + T_2^T S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i (D_1^i C_o D_o^i)^T \end{aligned} \quad (2.67)$$

where

$$\mathcal{X}_{WS}^i(t_f, C_o) = \int_0^{t_f} e^{(F^i + \alpha^i I)t} \Gamma^i W_o^i \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} e^{(F^i + \alpha^i I)^T t} dt \quad (2.68)$$

$$\begin{aligned} \mathcal{M}_{WS}^i(t_f, C_o) &= \int_0^{t_f} \int_0^t e^{(F^i + \alpha^i I)^T (t-\sigma)} \left( H_c^{iT} Q^i H_c^i + C^{iT} R^i C^i \right) e^{(F^i + \alpha^i I)t} \\ &\quad \Gamma^i W_o^i \Gamma^{iT} S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} e^{(F^i + \alpha^i I)^T \sigma} d\sigma dt \end{aligned} \quad (2.69)$$

#### 2.5.5 Case $D_cD_{su} = 0$

Using the definitions of  $F^i$ ,  $\Gamma^i$ ,  $H_c^i$ , and  $C^i$  given in equations (2.61)-(2.64), we have

$$\begin{aligned} \frac{\partial \tilde{J}_\sigma(t_f, C_o)}{\partial C_o} &= \sum_{i=1}^{N_p} W_p^i \left\{ D_2^{iT} Q^i H_c^i + T_1^T R^i C^i \right\} \mathcal{X}_{WS}^i(t_f, C_o) (H_o^i + D_1^i C_o T_3)^T \\ &\quad + D_1^{iT} C_o^T \left\{ D_2^{iT} Q^i H_c^i + T_1^T R^i C^i \right\} \mathcal{X}_{WS}^i(t_f, C_o) T_3^T \\ &\quad + G_o^{iT} \mathcal{M}_{WS}^i(t_f, C_o) [H_o^i + D_1^i C_o T_3]^T + (G_o^i C_o D_1^i)^T \mathcal{M}_{WS}^i(t_f, C_o) T_3^T \\ &\quad + G_o^{iT} S^i(t_f, C_o) \Gamma^i W_o^i \Gamma^{iT} S^i(t_f, C_o) [\Gamma_o^i + G_o^i C_o D_o^i] W_o^i D_o^{iT} \end{aligned} \quad (2.70)$$

## 2.6 Special Design Problems

For further study, we simplify the problem described in the previous section to a single nominal plant condition (i.e.,  $N_p = 1$ ). The plant state model is given by equations (2.1)-(2.3) and has the following simplified form

$$\begin{aligned} \dot{x}(t) &= Fx(t) + Gu(t) && \text{(Plant dynamics)} \\ x(0) &= x_0 && \text{(Initial conditions)} \\ y_s(t) &= H_s x(t) && \text{(Measurement variables)} \\ z(t) &= H_c x(t) && \text{(Criterion variables)} \end{aligned}$$

where we omit the superscript  $i$  (i.e.,  $i = 1$ ) for clarity. Note that simplification has been made to the measurement and criterion equations where direct influence from the control and disturbance inputs has been omitted (i.e.,  $D_{su} = D_{sw} = D_{cu} = D_{cw} = 0$ ). For a deterministic control problem related to plant initial conditions, the disturbance inputs  $w(t)$  are not considered in the controller synthesis. Furthermore, it is assumed that the pair  $(F, G)$  is controllable,  $(F, H_s)$  is observable, and  $(F, H_c)$  is detectable.

In the conventional LQR problem, one seeks a control-law  $u(t)$  that minimizes the following cost functional

$$J(t_f, u(t)) = \frac{1}{2} \int_0^{t_f} \left[ z^T(t) Q z(t) + u^T(t) R u(t) \right] dt \quad (2.71)$$

where the weighting matrices  $Q$  and  $R$  are both symmetric and positive definite. Solution to this dynamic optimization problem would normally proceed by adjoining the state dynamics  $\dot{x}(t) = Fx(t) + Gu(t)$  as constraints to the cost functional  $J(t_f)$  using a set of Lagrange multipliers  $\lambda(t)$ . From the stationary condition  $\delta J(x, u, \lambda, t_f) = 0$ , we obtain the familiar Hamilton-Jacobi equations for a two-point boundary value problem. The resulting optimal control-law is given by  $u(t) = -R^{-1}G^T\lambda(t)$ . By defining  $\lambda(t) = S(t)x(t)$ , and letting  $t_f \rightarrow \infty$  for the steady-state problem, one obtains a stabilizing static state-feedback control-law  $u(t) = D_c x(t)$ , where the gain matrix  $D_c$  is given by  $D_c = -R^{-1}G^T S_{ss}$ , and where  $S_{ss}$  satisfies the algebraic Riccati equation

$$F^T S_{ss} + S_{ss} F - S_{ss} G R^{-1} G^T S_{ss} + H_c^T Q H_c = 0. \quad (2.72)$$

We now examine an alternative development to the above LQR problem in the perspective of direct optimization using a fixed controller design structure. The problem is formulated as detailed in Section 2.4 and simplified to the case of a *static* output-feedback design (i.e., with  $C_o = D_c$  or  $u(t) = D_c y_s(t)$ ). Clearly, the state-feedback case of LQR design is simply a special case where  $y_s(t) = Ix(t)$ , i.e.,  $H_s = I$ . Closed-loop system state responses to initial conditions under a static output-feedback law are given by

$$x(t) = e^{(F+GD_cH_s)t} x_o = e^{F't} x_o \quad (2.73)$$

The cost function  $J(t_f, u(t))$  in equation (2.71) can be rewritten as

$$\begin{aligned} J(t_f, D_c) &= \frac{1}{2} E \left[ x_o^T S(t_f, D_c) x_o \right] \\ &= \frac{1}{2} \text{tr} \left\{ S(t_f, D_c) E \left[ x_o x_o^T \right] \right\} \end{aligned} \quad (2.74)$$

with  $S(t_f, D_c)$  is given in equation (2.31), and the initial condition vector  $x_o$  can be treated as a set of random variables with zero mean  $E[x_o] = 0$  and covariance  $E[x_o x_o^T] = W_o \geq 0$ .

The control design problem reduces then to the minimization of  $J(t_f, D_c)$  with respect to the static output-feedback gain matrix  $D_c$ . That is,

$$\frac{\partial J(t_f, D_c)}{\partial D_c} = \frac{1}{2} \frac{\partial}{\partial D_c} \text{tr} \{ S(t_f, D_c) W_o \} \quad (2.75)$$

Using Kleinman's lemma [32] and after some manipulation, we obtain

$$\begin{aligned} \frac{\partial J(t_f, D_c)}{\partial D_c} &= G^T \int_0^{t_f} S(t_f - \sigma, D_c) e^{(F^i + \alpha^i I)\sigma} W_o e^{(F^i + \alpha^i I)^T \sigma} d\sigma H_s^T \\ &\quad + R D_c H_s \int_0^{t_f} e^{(F^i + \alpha^i I)\tau} W_o e^{(F^i + \alpha^i I)^T \tau} d\tau H_s^T \end{aligned} \quad (2.76)$$

One could also obtain the above result using the general gradient expressions given in equations (2.57) or (2.65) when we specialize to the case of a single plant condition and  $C_o = D_c$ .

Solution to the necessary condition for optimality

$$\frac{\partial J(t_f, D_c)}{\partial D_c} = 0 \quad (2.77)$$

for the optimal gain matrix  $D_c$  is difficult when the terminal time  $t_f$  is finite. However, for a stable closed-loop system matrix  $F'$  and as  $t_f$  is increased to beyond five or six times the time constant of the slowest closed-loop modes, then

$$\int_0^{t_f} S(t_f - \sigma, D_c) e^{F'\sigma} W_o e^{F'^T\sigma} d\sigma \rightarrow S(t_f, D_c) \left\{ \int_0^{t_f} e^{F'\sigma} W_o e^{F'^T\sigma} d\sigma \right\} \quad (2.78)$$

Under this *steady-state* condition, we can re-assemble equation (2.76) as

$$\left( R D_c H_s + G^T S(t_f, D_c) \right) \left\{ \int_0^{t_f} e^{(F' + \alpha^i I)\sigma} W_o e^{(F' + \alpha^i I)^T\sigma} d\sigma \right\} H_s^T \simeq 0 \quad (2.79)$$

or, the term  $\left( H_s^T D_c^T R + S(t_f, D_c) G \right)$  is in the null space of the matrix

$$H_s \int_0^{t_f} e^{(F' + \alpha^i I)\sigma} W_o e^{(F' + \alpha^i I)^T\sigma} d\sigma$$

For the special case where  $H_s = I$  (i.e., the state-feedback case) and  $W_o > 0$ , we obtain the familiar result of

$$D_c \simeq -R^{-1} G^T S(t_f, D_c) \quad (2.80)$$

To show the correspondence between the matrix  $S(t_f, D_c)$  and the Riccati equation in the state-feedback case, we substitute equation (2.80) into equation (2.31) to obtain

$$S(t_f, D_c) \simeq \int_0^{t_f} e^{(F - GR^{-1}G^T S(t_f, D_c))T} \left( H_c^T Q H_c + S(t_f, D_c) G R^{-1} G^T S(t_f, D_c) \right) e^{(F - GR^{-1}G^T S(t_f, D_c))t} dt \quad (2.81)$$

Multiplying the above equation by  $(F^T - S(t_f, D_c) G R^{-1} G^T)$  on the left and adding it to the same equation multiplied by  $(F - GR^{-1}G^T S(t_f, D_c))$  on the right, we obtain

$$\begin{aligned} & \left[ F^T - S(t_f, D_c) G R^{-1} G^T \right] S(t_f, D_c) + S(t_f, D_c) \left[ F - GR^{-1}G^T S(t_f, D_c) \right] = \\ & e^{(F - GR^{-1}G^T S(t_f, D_c))T} \left[ H_c^T Q H_c + S(t_f, D_c) G R^{-1} G^T S(t_f, D_c) \right] e^{(F - GR^{-1}G^T S(t_f, D_c))t} \Big|_0^{t_f} \end{aligned} \quad (2.82)$$

or

$$\begin{aligned} & F^T S(t_f, D_c) + S(t_f, D_c) F - S(t_f, D_c) G R^{-1} G^T S(t_f, D_c) + H_c^T Q H_c = \\ & e^{(F - GR^{-1}G^T S(t_f, D_c))T} \left[ H_c^T Q H_c + S(t_f, D_c) G R^{-1} G^T S(t_f, D_c) \right] e^{(F - GR^{-1}G^T S(t_f, D_c))t} \Big|_0^{t_f} \end{aligned} \quad (2.83)$$

The above equation reduces to the algebraic Riccati equation under a stabilizing state feedback design when  $t_f \rightarrow \infty$ ,

$$F^T S_{ss} + S_{ss} F - S_{ss} G R^{-1} G^T S_{ss} + H_c^T Q H_c = 0, \quad (2.84)$$

where

$$S_{ss} = \lim_{t_f \rightarrow \infty} S(t_f, D_c). \quad (2.85)$$

Output feedback designs formulated in this manner yield the minimum cost functional value over an expected distribution of initial conditions. Often, as in the conventional state-feedback design, this expected distribution is assumed independent and uniform (i.e.,  $E[x_o x_o^T] = I$ ).

In the static output-feedback design case (where  $H_s \neq I$ ), analytical closed-form solutions cannot be found in general for the design problem stated in equation (2.28). One must resort to numerical techniques for the solution of the optimal design gain matrix  $D_c$ . Note that, in the above formulation, solution uniqueness as well as the optimal solution itself will depend on the selection of the initial condition covariance matrix  $W_o$ . Recently, Chen, et al. [33] have provided precise conditions under which an optimal static state-feedback design is unique. Non-uniqueness of an optimal state-feedback design can be used to explore other design issues such as robustness ([37]-[38]).

One can also apply this special design case to the approximate worst-case cost. The stationary conditions given by

$$\frac{\partial \tilde{J}_{\tilde{\sigma}}}{\partial D_c} = \frac{\partial}{\partial D_c} \text{tr} \left\{ \left( S(t_f, D_c) \Gamma W_o \Gamma^T \right)^2 \right\} = 0$$

become, for a sufficiently large  $t_f$ ,

$$0 \simeq \left( R D_c H_s + G^T S(t_f, D_c) \right) \Phi^*(t_f) H_s^T \quad (2.86)$$

where

$$\Phi^*(t_f) = \int_0^{t_f} e^{(F'^i + \alpha^i I)\sigma} \Gamma W_o \Gamma^T S(t_f, D_c) \Gamma W_o \Gamma^T e^{(F'^i + \alpha^i I)^T \sigma} d\sigma$$

One possible solution is given by the relation  $D_c H_s \simeq -R^{-1} B^T S(t_f, D_c)$ . Note that the above equation is identical to the  $H^2$  form except for the weighting matrix

$$\left\{ \int_0^{t_f} e^{(F'^i + \alpha^i I)\sigma} \Gamma W_o \Gamma^T S(t_f, D_c) \Gamma W_o \Gamma^T e^{(F'^i + \alpha^i I)^T \sigma} d\sigma \right\} H_s^T.$$

This is significant in light of the fact that a dynamic output feedback problem can also be reformulated in terms of static output feedback [2] for linear time-invariant systems, thus one can show that a similar weighting factor is responsible for the difference in  $H^2$  and worst-case solutions for the dynamic output feedback case as well. Clearly, this solution is identical to the solution of the LQR problem when  $H_s = I$ . Generally, solutions of the above stationary condition for the optimal static gain matrix  $D_c$  can be only be done using numerical optimization.

## 2.7 Design Example Using a Simplified Helicopter Model

A comparison of designs achieved under the LQ,  $H^2$ , and worst-case performance index is done for the case of state feedback. The state feedback result also establishes a baseline for the output-feedback designs.

For output feedback, we compare designs from three different performance objectives: LQG,  $H^2$ , and worst-case. Because the disturbance set-up in the synthesis model is so

important, this comparison is performed over 2 sets of synthesis models. The first set is related to loop transfer recovery (LTR). The second set involves introducing fictitious noises into each of the states of the plant model with different intensity levels. This is an attempt to approach state-feedback results by improving the stability of the closed-loop system.

### 2.7.1 Helicopter Model

We consider a 1-dimensional model of the OH6A helicopter in hover mode (see Ref. [19]) with a longitudinal cyclic control  $\delta(t)$ . The vehicle states are pitch rate  $q(t)$ , pitch angle  $\theta(t)$ , ground velocity  $u(t)$ , and position  $x(t)$ . The system units are feet, seconds, and degrees. The state model is

$$\begin{bmatrix} \dot{u}(t) \\ \dot{q}(t) \\ \dot{\theta}(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} X_u & X_q & -g & 0 \\ M_u & M_q & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u(t) \\ q(t) \\ \theta(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} X_\delta \\ M_\delta \\ 0 \\ 0 \end{bmatrix} \delta(t) + \begin{bmatrix} -X_u \\ -M_u \\ 0 \\ 0 \end{bmatrix} u_w(t). \quad (2.87)$$

Values of the constants in the model matrices are  $X_u = -0.0257$ ,  $X_q = 0.013$ ,  $M_u = 1.26$ ,  $M_q = -1.765$ ,  $X_\delta = 0.086$ , and  $M_\delta = -7.4080$ .

The vehicle is disturbed by a horizontal gust  $u_w(t)$ . It is modeled as a white noise with a spectral density of 18 ft<sup>2</sup>/sec and entered in all designs as well as in the calculation the RMS gust responses.

### 2.7.2 Full-State Feedback Design

Initially we consider a full state-feedback controller to minimize the following quadratic cost objective

$$J(t_f, D_c) = \int_0^{t_f} E \left[ x^2(t) + \delta^2(t) \right] dt \quad (2.88)$$

which penalizes the ground position  $x(t)$  and the longitudinal cyclic control  $\delta(t)$  equally. The optimum design  $D_c$  is found to be  $D_c = [1.9890, -0.2560, -0.7589, 1.00]$ . This solution can be obtained both from the Riccati equation (2.84) and from direct optimization based on equation (2.76). In the gradient search procedure, the function *nelder* in *MATLAB* was used to minimize the following objective functions: the objective function  $J(t_f, D_c)$  in equation (2.88), the maximum singular value  $J_\sigma$  for the exact worst-case design defined in equation (2.38), and the Frobenius norm for the approximate worst-case design in equation (2.39). In *all* numerical cases, the terminal time  $t_f$  is set equal to 512 seconds, which meets the aforementioned criterion for being steady-state.

Table 2.1: Static State Feedback for the Helicopter Model

Eigenvalues		Single-Loop Margins			Unit Gust Responses
Open-Loop	Closed-Loop	Loop	Phase	Gain	
-1.8891	-1.8461	$x$ :	60.01°	8.61dB	$x = 0.243$
0	-1.1192	$\theta$ :	41.4°	-5.85dB	$\theta = 1.497$
0.0492 ± 0.4608i	-0.4464 ± 0.9774i	$\delta$ :	60.2°	-11.3dB	$\delta = 0.976$
( $\zeta = -0.106$ )	( $\zeta = 0.415$ )				

Design results are shown in Table 2.1. They provide the baseline for comparison with the output feedback designs that follow. Time responses to a step command in ground

position,  $x_{cmd}$ , are shown in Figure 2.2 along with the Bode magnitude plot of the transfer function  $x(s)/x_{cmd}(s)$ .

Design solutions are identical in all three cases and are equal to those obtained from the Riccati equation for the LQ problem. Note again that the LQ design is equivalent to the minimization of an objective function evaluated to a uniformly distributed set of initial conditions on all four rotorcraft states, i.e., with  $E[x_0 x_0^T] = I$ . In the numerical optimization, the initial conditions enter into the evaluation of the different types of cost functions explicitly. In the minimization of  $tr\{S(t_f, D_c)\}$  for LQ and  $H^2$  problems,  $\bar{\sigma}(S(t_f, D_c))$  in worst-case, and  $tr\{S^2(t_f, D_c)\}$  in the approximate worst case, the maximum singular value of  $S(t_f, D_c)$  is found to be 5.58.

### 2.7.3 Comparison of Output-Feedback Designs

To make a valid comparison, the controller order must be the same for each of the LQG,  $H^2$ , and worst-case designs. Since the LQG design results in a 4th order controller with no direct feedthrough terms, all other designs based on direct optimization will involve a 4th-order controller with  $D_c$  set to zero.

The LQG formulation is set as to the form of the disturbance inputs it uses, though being an observer-based design, it actually marries two different disturbance models. The state-feedback gains were determined using a unit uniform uncorrelated distribution on the initial conditions of all the rotorcraft states. The full-order observer part of the LQG design assumes white noise for both process and sensors. The  $H^2$  and worst-case problems will also be formulated with white noise. We define the sensor outputs for the rotorcraft as the rotorcraft position  $x$  and pitch angle  $\theta$ . Sensor noise in the ground position measurement  $x(t)$  has a spectral density of  $0.4 \text{ ft}^2 \text{ sec}$ , and for the pitch measurement  $\theta(t)$  a sensor noise of  $0.4 \text{ deg}^2 \text{ sec}$ . The helicopter is excited by a horizontal gust of spectral density  $18 \text{ ft}^2/\text{sec}$ .

An initial output-feedback design is performed using only the gust disturbance as process noise. Compared to the results in Table 2.1 for state feedback, the RMS responses to the gust input are higher in all three output-feedback designs (Table 2.2). Robustness evaluation is conducted at the control actuator input. The resulting gain and phase margins are low. The maximum singular value of  $S$  in the LQG design is somewhat larger than that for the state feedback design, and those from the  $H^2$  and worst-case designs are significantly higher still. The approximate worst-case design does indeed optimize to a smaller  $\bar{\sigma}(\Gamma^T S \Gamma)$  than the other designs. This trend is maintained for the different sets of disturbances. Note that “tuning” of the worst-case design to gust disturbances leads to a poor step response characteristic. (Figure 2.4) Responses to command inputs for the LQG design always matches corresponding responses of the LQ design.

### 2.7.4 Actuator Disturbance Augmentation

Here we augment the disturbance model to include a process noise entering into the control actuator by letting  $\Gamma = [\Gamma_g \ \alpha G]$ . The power spectral density of this process noise is fixed at  $18 \text{ deg}^2 \text{ sec}$ . (Note that the gust disturbance power spectral density is  $18 \text{ ft}^2/\text{sec}$ .) The factor  $\alpha$  in the disturbance distribution matrix is used to scale the effects of this additional process noise relative to the gust disturbance, and has the sequence of values  $[0, 0.1, 1, 10]$ . This process simply follows the traditional Loop Transfer Recovery (LTR) procedure in LQG designs, to enhance the robustness in the actuator loop [39]. Table 2.2 shows the expected improvement in robustness. Responses to the gust disturbance also improved

Table 2.2: Traditional Loop Transfer Recovery Using Different Design Methods  
LQG Designs

Fictitious Actuator Noise Factor $\alpha$		0.0	0.1	1.0	10.0
Mean Square	x:	1.255	1.165	0.627	0.431
Responses	$\theta$ :	6.617	6.125	3.034	1.888
to $u_w$	$\delta$ :	1.274	1.254	1.069	0.950
x:	P.M. (deg)	31.66	31.56	30.83	30.51
	G.M. (db)	5.04	5.02	4.90	4.88
$\theta$ :	P.M. (deg)	23.93	24.93	30.30	30.09
	G.M. (db)	10.63,-4.110	10.95,-4.117	15.63,-3.569	-3.75
$\delta$ :	P.M. (deg)	29.84	29.90	40.62	52.49
	G.M. (db)	10.66,-7.46	10.97,-7.638	15.62,-9.265	-10.55
$\bar{\sigma}(S)$ :		28.57	28.81	30.66	31.56
$\bar{\sigma}(\Gamma'^T ST')$ :		3.882	4.442	53.48	4111

$H^2$  Numerical Designs

Fictitious Actuator Noise		0.0	0.1	1.0	10.0
Mean Square	x:	1.276	1.182	0.627	0.431
Responses	$\theta$ :	6.596	6.102	3.034	1.889
to $u_w$	$\delta$ :	1.269	1.244	1.069	0.950
x:	P.M. (deg)	32.35	33.31	30.83	30.51
	G.M. (db)	5.15	5.15	4.91	4.88
$\theta$ :	P.M. (deg)	24.23	24.75	30.30	30.09
	G.M. (db)	10.45,-3.268	10.75,-3.296	15.61,-3.569	-3.75
$\delta$ :	P.M. (deg)	27.97	28.46	40.58	52.47
	G.M. (db)	10.49,-7.86	10.78,-7.940	15.60,-9.260	-10.55
$\bar{\sigma}(S)$ :		12138	20527	2263.	725.3
$\bar{\sigma}(\Gamma'^T ST')$ :		3.889	4.447	53.52	4112

Worst-Case Numerical Designs

Fictitious Actuator Noise		0.0	0.1	1.0	10.0
Mean Square	x:	0.948	0.887	0.529	0.342
Responses	$\theta$ :	6.887	6.209	2.798	1.901
to $u_w$	$\delta$ :	1.398	1.343	1.036	0.965
x:	P.M. (deg)	29.24	29.78	30.08	30.72
	G.M. (db)	4.03	4.01	4.10	5.31
$\theta$ :	P.M. (deg)	24.75	23.56	29.40	30.36
	G.M. (db)	11.46,-2.828	12.03,-2.785	18.86,-2.981	-3.57
$\delta$ :	P.M. (deg)	29.67	27.53	43.37	55.00
	G.M. (db)	11.49,-6.18	12.04,-6.514	18.87,-9.495	-10.80
$\bar{\sigma}(S)$ :		9448.	16894	1840.	122.1
$\bar{\sigma}(\Gamma'^T ST')$ :		3.562	4.072	50.40	3972

when we increase the actuator process noise. Although there is no general tendency for  $\bar{\sigma}(S)$  to converge toward the state-feedback result, this value did decrease somewhat with increasing  $\alpha$  in the  $H^2$  and worst-case designs. For different values of  $\alpha$ , the LQG and  $H^2$  designs show significantly different results in  $\bar{\sigma}(S)$  and in step responses to  $x_{cmd}$ .

### 2.7.5 Fictitious State Noise Augmentation

In this set of designs, we augment the disturbance model with independent fictitious noises entering into each of the plant states by defining  $\Gamma = [\Gamma_g \ \alpha I]$ . We set the power spectral density of each fictitious noise at 18. Again the scaling factor  $\alpha$  in the disturbance distribution matrix takes on a sequence of values [0,0.1,1,10]. The ensuing stability augmentation improves the overall controller performance, with the parameter  $\bar{\sigma}(S)$ , representative of the state-feedback performance, trending toward the state-feedback value of 5.58 as  $\alpha$  is increased.

In the LQG design, increasing the process noises in each state will cause the estimated states of the observer to converge more rapidly. As a result, robustness at the actuator loop worsens, while the maximum singular value of  $S$  approaches that of the state feedback design (Table 2.3). The single-loop robustness at the sensors improves, and the controller is less attuned to gust alleviation in the position  $x$  (as in the  $H^2$ -based controller designs). The step responses and the transfer functions  $x/x_{cmd}$  are equal to those of state feedback as expected.

Results based on the numerical optimization of  $H^2$  costs do not correspond to those of LQG as well as in the previous case, though overall they have similar gust responses and robustness margins. Increasing the relative emphasis of fictitious disturbances going into the plant states caused no convergence of  $H^2$  and LQG design properties.

The step responses in the  $H^2$  designs (Figure 2.5) converge nearly to the state-feedback response as  $\alpha$  is increased, though with a slightly slower response and less overshoot. While  $\bar{\sigma}(S)$  became smaller, did not settle to the state feedback value.

For the worst-case designs, increasing the fictitious plant state noises slightly improved the overall disturbance rejection, while the robustness for all loops improved more substantially. Of note is that the robustness at the actuator loop improved. The value of  $\bar{\sigma}(S)$  tends towards that of the state feedback case, while the step responses (Figure 2.6) also converge to nearly the state feedback response. However, in spite of increased system stability, disturbing the initial conditions of the states did not result in as desirable overall controller characteristics as augmenting with actuator disturbances. More favorable results may be obtained with larger disturbance amplitudes.

## 2.8 Conclusions

Direct parameter optimization provides a powerful means to address a wide class of design criteria and controller structures. The control formulation presented in this chapter shows several common types that are well-suited for the framework of numerical optimization. Development of reliable numerical optimization techniques is tied closely to the availability of reliable computation of the cost functions and its gradients with respect to the controller design variables. In all cases, evaluation of the cost functions presented in this chapter and their gradients involves directly the computation of the following integrals of matrix

Table 2.3: Singular Value Type Disturbance Augmentation  
LQG Designs

Fictitious State Noise Factor $\alpha$		0.0	0.1	1.0	10.0
Mean Square	x:	1.255	1.410	2.653	2.674
Responses	$\theta$ :	6.617	6.769	9.130	8.913
to $u_w$	$\delta$ :	1.274	1.215	1.072	1.017
x:	P.M. (deg)	31.66	30.96	34.77	35.76
	G.M. (db)	5.04	4.44	5.83	6.58
$\theta$ :	P.M. (deg)	23.93	22.76	25.60	28.03
	G.M. (db)	10.63,-4.110	10.91,-2.806	12.83,-3.686	15.66,-4.06
$\delta$ :	P.M. (deg)	29.84	27.87	25.30	25.60
	G.M. (db)	10.66,-7.460	10.97,-8.10	12.84,-8.516	14.41,-8.63
	$\bar{\sigma}(S)$ :	28.57	16.36	8.450	7.553
	$\bar{\sigma}(\Gamma'^T S \Gamma')$ :	3.882	5.647	126.0	11140

$H^2$  Numerical Designs

Fictitious State Noise Factor $\alpha$		0.0	0.1	1.0	10.0
Mean Square	x:	1.276	1.488	2.656	2.053
Responses	$\theta$ :	6.596	7.082	9.049	7.810
to $u_w$	$\delta$ :	1.269	1.187	1.064	1.040
x:	P.M. (deg)	32.35	32.56	34.81	34.84
	G.M. (db)	5.15	4.66	5.83	6.20
$\theta$ :	P.M. (deg)	24.23	23.64	25.82	27.79
	G.M. (db)	10.45,-3.268	11.25,-2.839	12.90,-3.698	16.22,-3.75
$\delta$ :	P.M. (deg)	27.97	28.83	25.58	26.65
	G.M. (db)	10.49,-7.86	11.33,-7.703	12.40,-8.562	16.02,-8.60
	$\bar{\sigma}(S)$ :	12137	1038.	4300.	70.11
	$\bar{\sigma}(\Gamma'^T S \Gamma')$ :	3.889	5.690	126.1	11101

Worst-Case Numerical Designs

Fictitious State Noise Factor $\alpha$		0.0	0.1	1.0	10.0
Mean Square	x:	0.948	0.777	1.597	1.432
Responses	$\theta$ :	6.887	5.714	6.720	6.090
to $u_w$	$\delta$ :	1.398	1.403	1.049	0.967
x:	P.M. (deg)	29.24	29.20	32.71	33.33
	G.M. (db)	4.03	3.96	5.96	6.26
$\theta$ :	P.M. (deg)	24.75	22.97	28.69	30.93
	G.M. (db)	11.46,-2.828	11.40,-2.623	16.82,-3.637	-3.79
$\delta$ :	P.M. (deg)	29.67	27.79	28.88	29.86
	G.M. (db)	11.49,-6.18	11.43,-6.977	16.70,-8.596	-8.87
	$\bar{\sigma}(S)$ :	9448.	6770.	296.0	9.523
	$\bar{\sigma}(\Gamma'^T S \Gamma')$ :	3.562	5.221	113.1	10413

exponentials

$$\mathcal{X}(t) = \int_0^t e^{A\tau} B e^{C\tau} d\tau$$

and

$$\mathcal{M}(t) = \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv.$$

Note that the matrix  $S(t)$  has the same form as  $\mathcal{X}(t)$ . Efficient algorithms have already been developed for the above integrals based on diagonalization of the system matrix. However, they are prone to inaccuracies and lead to convergence problems in numerical optimization when the system matrix contains defective degenerate modes. A reliable algorithm for evaluating these integrals has been developed and its details are given in Section 3.3.

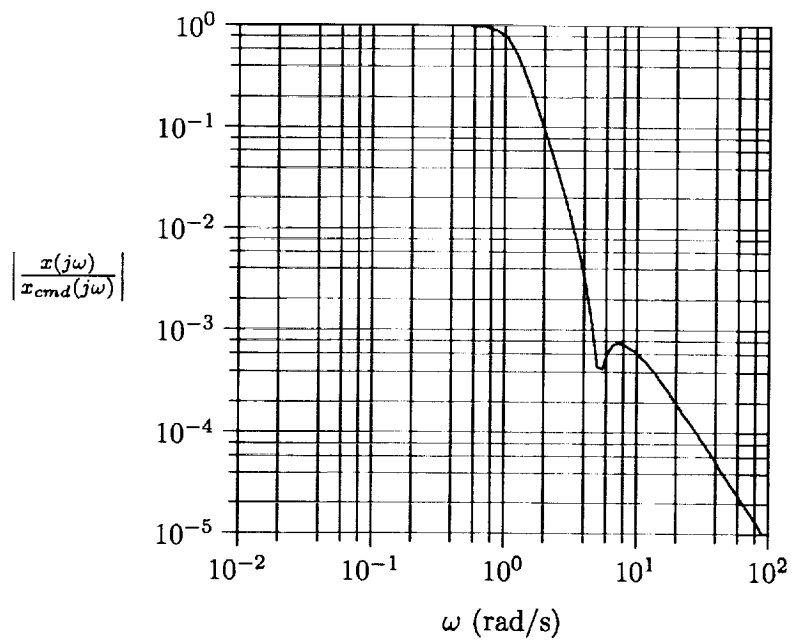
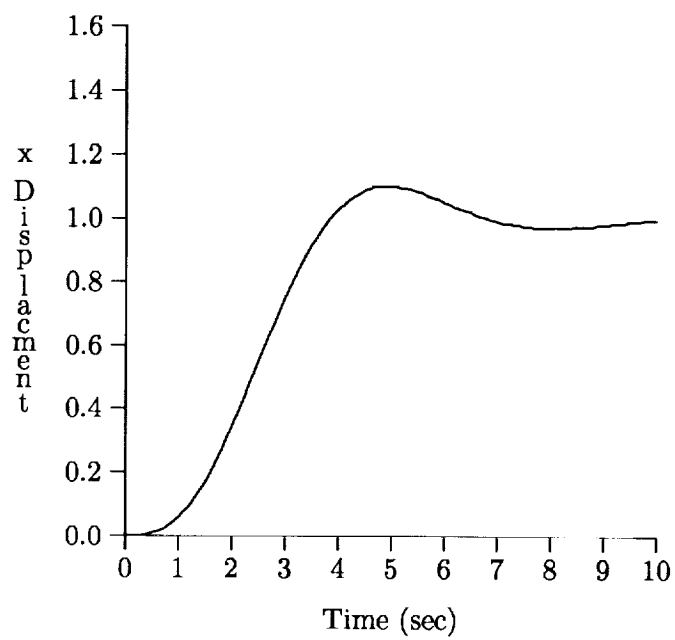


Figure 2.2: State-Feedback Design for the Helicopter Model

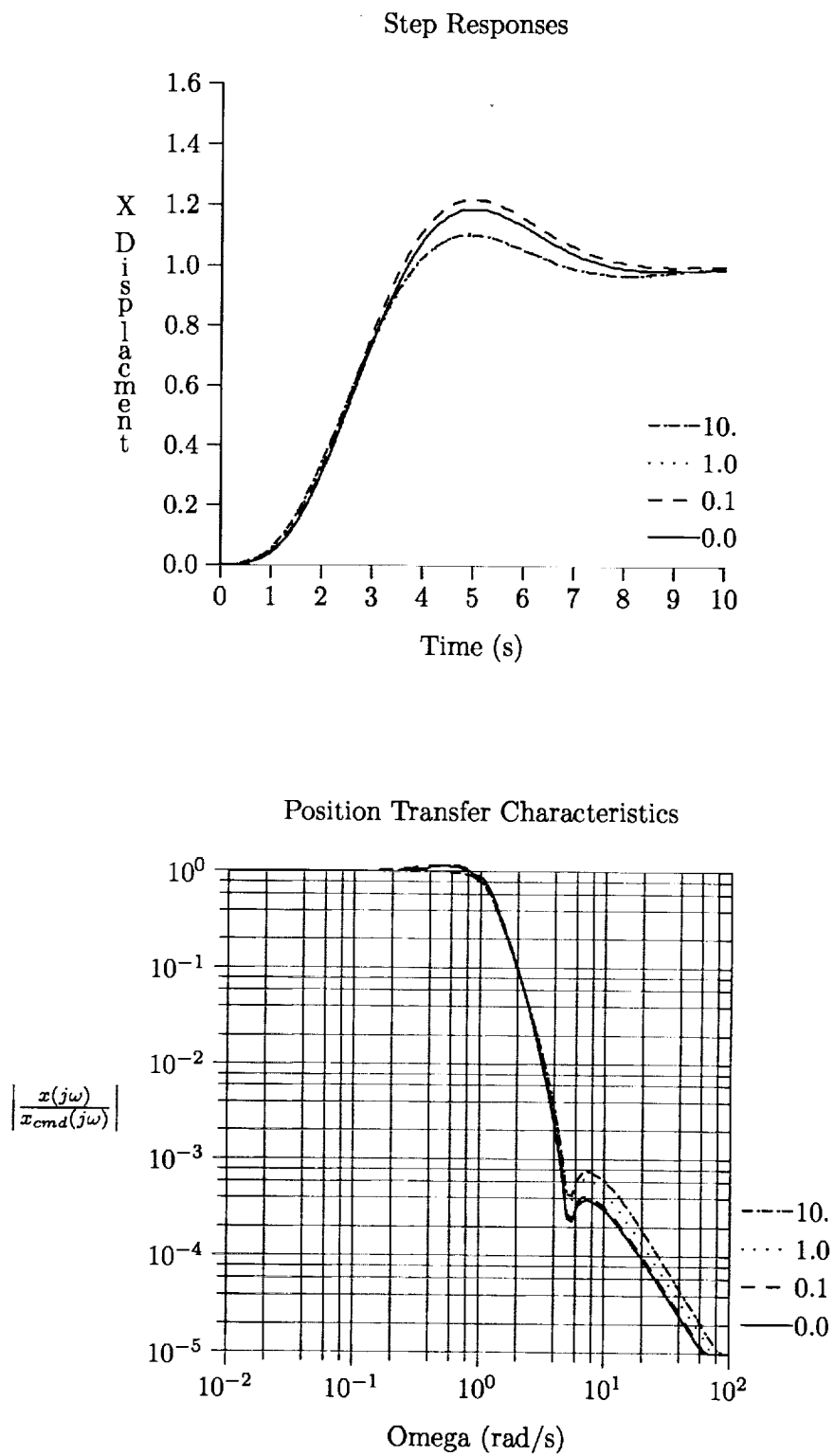


Figure 2.3:  $H^2$  Designs for Various Loop Recovery Weights

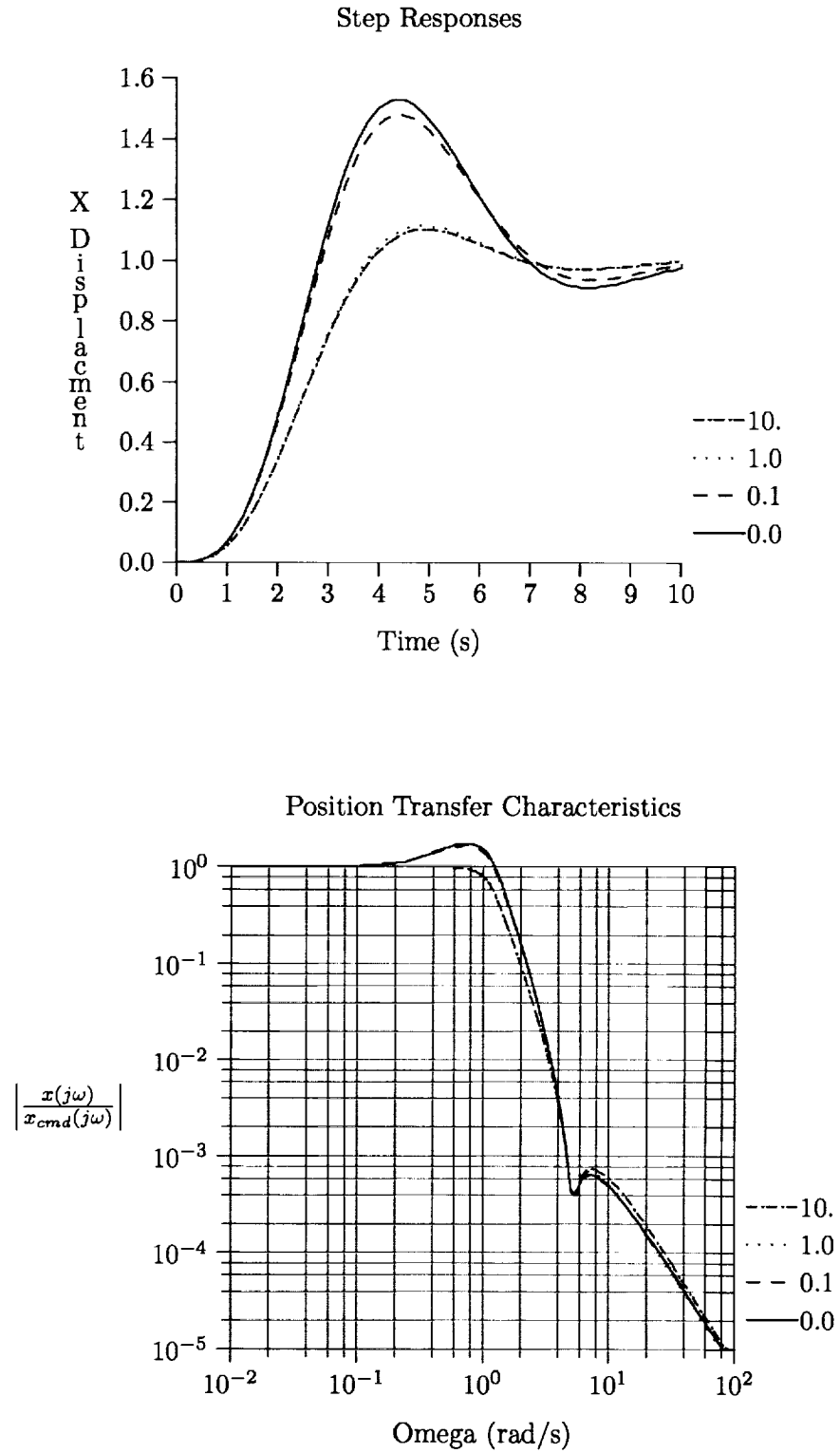


Figure 2.4: Worst-Case Designs for Various Loop Recovery Weights

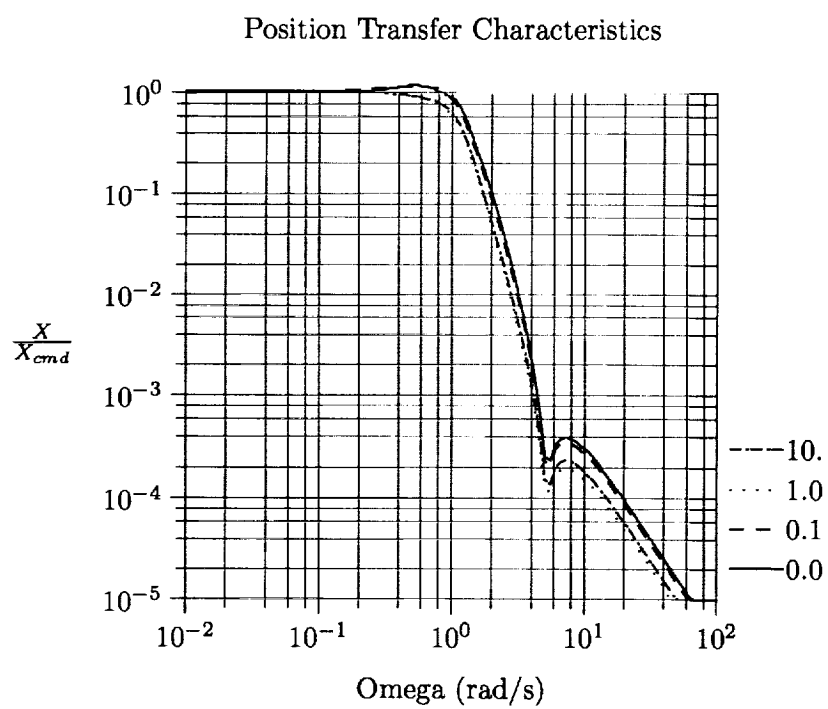
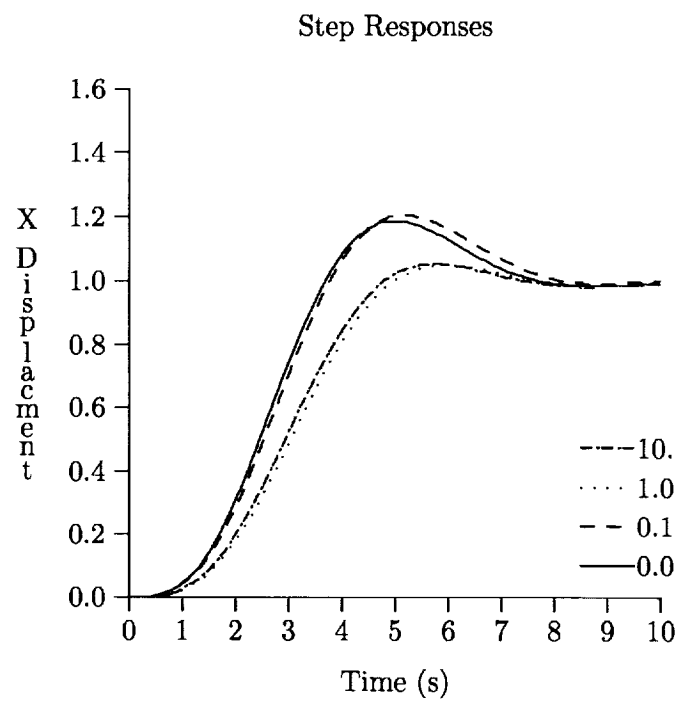


Figure 2.5:  $H^2$  Designs for Various Stability Weights

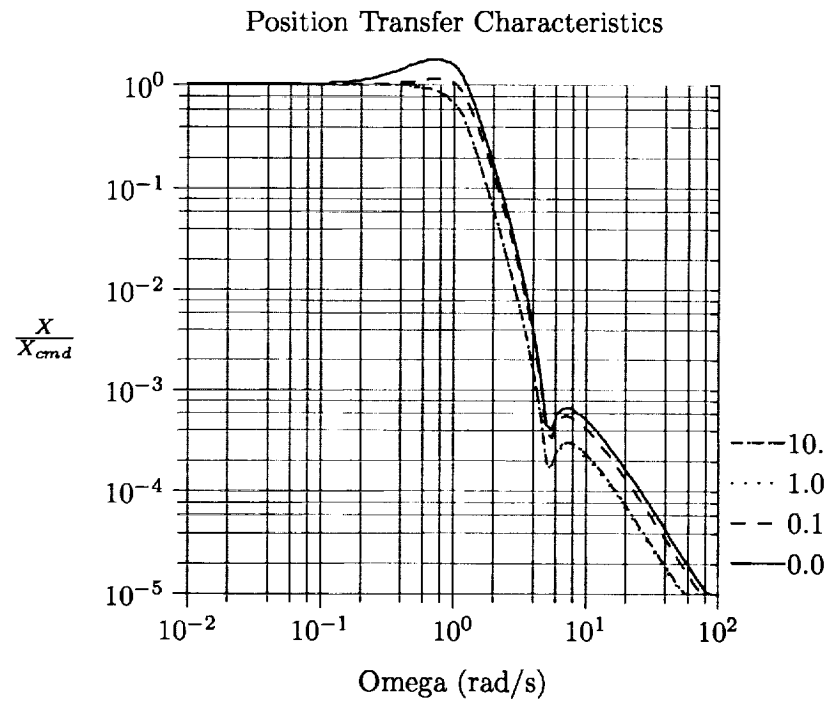
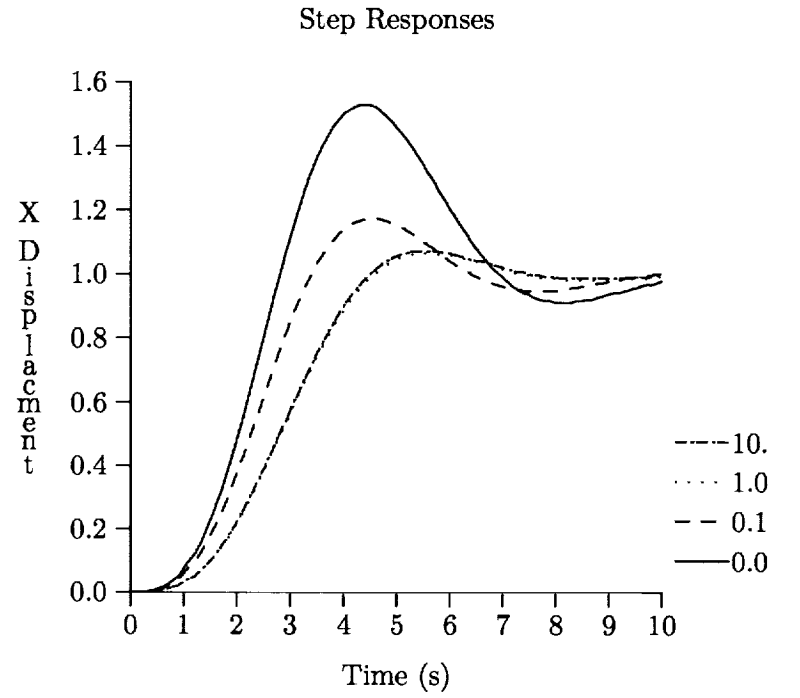


Figure 2.6: Worst-Case Designs for Various Stability Weights



## Chapter 3

# Evaluating $\mathcal{X}(t)$ and $\mathcal{M}(t)$

### 3.1 Introduction

Established methods for evaluating  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$  are based on diagonalization of the system matrices  $A$ ,  $C$  and  $E$  in the exponential functions. Diagonalization is achieved using a similarity transformation derived from the eigenvalue-eigenvector decomposition of these matrices. It is further assumed that the similarity transformation constructed from the eigenvector matrix is nonsingular. Let  $V_A$ ,  $V_C$  and  $V_E$  be the eigenvector matrices of the matrix  $A$ ,  $C$  and  $E$  respectively, then

$$A = V_A \Lambda_A V_A^{-1}, C = V_C \Lambda_C V_C^{-1}, E = V_E \Lambda_E V_E^{-1}, \quad (3.1)$$

where  $\Lambda_A$ ,  $\Lambda_C$  and  $\Lambda_E$  are diagonal matrices. One can express the exponential function of  $e^{At}$  as

$$e^{At} = e^{V_A \Lambda_A V_A^{-1} t} = V_A e^{\Lambda_A t} V_A^{-1}. \quad (3.2)$$

Application of this decomposition in the calculation of  $\mathcal{X}(t)$  is shown below.

$$\begin{aligned} \mathcal{X}(t) &= \int_0^t e^{A\tau} B e^{C\tau} d\tau \\ &= V_A \left\{ \int_0^t e^{\Lambda_A \tau} B e^{\Lambda_C \tau} d\tau \right\} V_C^{-1} \end{aligned} \quad (3.3)$$

where  $B = V_A^{-1} B V_C$ . Advantage of this approach is that the exponential function of a diagonal matrix is also diagonal. In this case, time integration in  $\mathcal{X}(t)$  can be performed directly by explicit integration of a product of scalar exponential functions. The resulting numerical algorithm is quite accurate and efficient, provided that the transformation matrices  $V_A$  and  $V_C$  are not ill-conditioned. A similar procedure can also be applied to the evaluation of  $\mathcal{M}(t)$ . Complete discussion can be found in Appendices C and D of [18]. Breakdown of this algorithm will occur when the matrices  $A$  or  $C$  develop Jordan blocks. This situation happens frequently in the control synthesis of flexible structures with densely packed modes as demonstrated in the design example of Section 4.2.

Clearly, in order to have a reliable design algorithm for optimal low-order output-feedback control synthesis [18], one must develop a robust numerical scheme to evaluate matrix integrals of the form shown in  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$  in the case where the system has defective eigenvalues.

### 3.2 Alternative Approaches for Solving $\mathcal{X}(t)$ and $\mathcal{M}(t)$

One simple approach is to evaluate

$$\mathcal{X}(t) = \int_0^t e^{A\tau} B e^{C\tau} d\tau, \quad \mathcal{M}(t) = \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv \quad (3.4)$$

directly using a numerical quadrature. Efficiency of numerical integration techniques is poor; especially when it requires small integration step size for satisfactory accuracy in the case of stiff system matrices  $A$ ,  $C$  and  $E$ . Another possibility is to use some types of algebraic Lyapunov equations for the solution of  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$ . For example, it can be easily shown that the matrix  $\mathcal{X}(t)$  can be obtained from the solution of the following Lyapunov equation,

$$A\mathcal{X}(t) + \mathcal{X}(t)C = \left[ e^{A\tau} B e^{C\tau} \right]_0^t \quad (3.5)$$

Solution of equation (3.5) exists if  $\lambda_i(A) + \lambda_j(C) \neq 0$ . As  $\lambda_i(A) + \lambda_j(C)$  tends toward zero, the solution accuracy will degrade on a continuous basis. Furthermore, there are many situations encountered in practice where this scheme will run into difficulty. For example, when  $A = C^T$  and the system matrix has poles at the origin. Thus, for practical purposes  $\mathcal{X}(t)$  cannot be solved from a scheme based on Lyapunov equations.

The Lyapunov equation for  $\mathcal{M}(t)$  is even more poorly behaved. By changing the order of integration, the double integral in  $\mathcal{M}$

$$\mathcal{M} = \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv$$

can be re-written as

$$\mathcal{M} = \int_0^t \left\{ \int_0^q e^{A(q-r)} B e^{-Cr} dr \right\} e^{Ct} D e^{E(t-q)} dq$$

Because the matrices  $A$ ,  $B$ , and  $C$  are time invariant, one can rewrite the convolution integral as

$$\int_0^t e^{A\tau} B e^{C(\tau-t)} d\tau = [\mathbf{I} \ 0] \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} t \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix}$$

Applying this to the inner integral of  $\mathcal{M}$ , we obtain

$$\mathcal{M} = \int_0^t \left\{ [\mathbf{I} \ 0] \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} q \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{E(t-q)} dq$$

This form is used to derive the Lyapunov equation for  $\mathcal{M}$ .

With the above equation, we write

$$\begin{aligned} \mathcal{M}E &= \int_0^t \left\{ [\mathbf{I} \ 0] \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} (t-q) \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{Eq} E dq \\ &= \left[ \left\{ [\mathbf{I} \ 0] \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} (t-q) \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{Eq} \right]_0^t \\ &\quad + \int_0^t \left\{ [\mathbf{I} \ 0] \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} (t-q) \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{Eq} E dq \end{aligned}$$

$$\begin{aligned}
&= \left[ 0 - \left\{ \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} t \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D \mathbf{I} \right] \\
&\quad + \int_0^t \left\{ \begin{bmatrix} A & B \end{bmatrix} \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} (t-q) \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{Eq} E dq
\end{aligned}$$

Using the relation  $\exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} t \right\} = \begin{bmatrix} e^{At} & \int_0^t e^{A(t-\tau)} B e^{-C\tau} d\tau \\ 0 & e^{-Ct} \end{bmatrix}$ , we have

$$\begin{aligned}
\mathcal{M}E &= - \left[ \left\{ \int_0^t e^{A(t-r)} B e^{-Cr} dr \right\} e^{Ct} D \right] + \int_0^t B e^{-C(t-q)} e^{Ct} D e^{Eq} dq \\
&\quad + \int_0^t A \left\{ \int_0^{t-q} e^{A((t-q)-r)} B e^{-Cr} dr \right\} e^{Ct} D e^{Eq} dq \\
&= B \int_0^t e^{-C(t-q)} e^{Ct} D e^{Eq} dq - \left[ \left\{ \int_0^t e^{A(t-r)} B e^{-Cr} dr \right\} e^{Ct} D \right] \\
&\quad + A \int_0^t \left\{ \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \exp \left\{ \begin{bmatrix} A & B \\ 0 & -C \end{bmatrix} (t-q) \right\} \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \right\} e^{Ct} D e^{Eq} E dq
\end{aligned}$$

Finally, we can assemble the Lyapunov equation for  $\mathcal{M}$  as

$$\mathcal{M}E - A\mathcal{M} = B \int_0^t e^{-C(t-q)} e^{Ct} D e^{Eq} dq - \left[ \left\{ \int_0^t e^{A(t-r)} B e^{-Cr} dr \right\} e^{Ct} D \right]$$

In most applications, the matrix  $E$  is equal to  $A$  and hence renders the Lyapunov equation for  $\mathcal{M}$  unsolvable.

### 3.3 Matrix Exponential Approach

Another possible approach is based on the direct use of the exponential of a matrix. It is well-known [23] that convolution integrals involving matrix exponentials, as represented in the matrices  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$ , can be derived from the matrix exponential of an augmented matrix. It can be shown that the matrix  $\mathcal{X}(t)$  can be derived from the upper-right partition of the following matrix exponential,

$$\mathcal{X}(t) = e^{At} \begin{bmatrix} I & 0 \end{bmatrix} \exp \left\{ \begin{bmatrix} -A & B \\ 0 & C \end{bmatrix} t \right\} \begin{bmatrix} 0 \\ I \end{bmatrix} \quad (3.6)$$

Thus, computation of  $\mathcal{X}(t)$  now involves the computation of a matrix exponential. A reliable algorithm for computing the matrix exponential is given in Section 3.4.

In a similar fashion, one can express the matrix  $\mathcal{M}(t)$  in terms of a submatrix of a matrix exponential. To see this, we start from its definition

$$\begin{aligned}
\mathcal{M}(t) &= \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv \\
&= \int_0^t e^{-As} \left\{ \int_s^t e^{Av} B e^{Cv} dv \right\} D e^{Es} ds \\
&= - \int_0^t e^{-As} \left\{ \int_t^s e^{Av} B e^{Cv} dv \right\} D e^{Es} ds
\end{aligned} \quad (3.7)$$

Let's perform a change of integration variable  $v = t - r$ . We have,

$$\begin{aligned}
\mathcal{M}(t) &= \int_0^t e^{-As} \left\{ \int_0^{t-s} e^{A(t-r)} B e^{C(t-r)} dr \right\} D e^{Es} ds \\
&= - \int_t^0 e^{A(t-s)} \left\{ \int_0^{t-s} e^{-Ar} B e^{-Cr} dr \right\} e^{Ct} D e^{-E(t-s)} d(t-s) e^{Et} \\
&= \int_0^t e^{Aq} \left\{ \int_0^q e^{-Ar} B e^{-Cr} dr \right\} e^{Ct} D e^{-Eq} dq e^{Et} \\
&= \int_0^t \left\{ \int_0^q e^{A(q-r)} B e^{Ct/2} e^{-Cr} dr \right\} e^{Ct/2} D e^{E(t-q)} dq
\end{aligned} \tag{3.8}$$

Notice that part of the integrand in equation (3.7) delimited by braces can be replaced by terms involving the exponential of an augmented matrix. This follows directly from results developed for the matrix  $\mathcal{X}(t)$ . With this substitution, we obtain

$$\begin{aligned}
\mathcal{M}(t) &= \int_0^t \left\{ [I \ 0] \exp \left\{ \begin{bmatrix} A & B e^{Ct/2} \\ 0 & -C \end{bmatrix} q \right\} \begin{bmatrix} 0 \\ I \end{bmatrix} \right\} e^{Ct/2} D e^{E(t-q)} dq \\
&= \int_0^t [I \ 0] \exp \left\{ \begin{bmatrix} A & B e^{Ct/2} \\ 0 & -C \end{bmatrix} (t-q) \right\} \left( \begin{bmatrix} 0 \\ I \end{bmatrix} e^{Ct/2} D \right) e^{Eq} dq \\
&= [I \ 0 \ 0] \exp \left\{ \begin{bmatrix} A & B e^{Ct/2} & 0 \\ 0 & -C & e^{Ct/2} D \\ 0 & 0 & E \end{bmatrix} t \right\} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}
\end{aligned} \tag{3.9}$$

In this section, we have shown that the matrices  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$  can be formulated in terms of solutions of matrix exponentials. Their evaluation depends therefore strongly on the accuracy and reliability of numerical methods for computing matrix exponential. We will present one such algorithm in Section 3.4. However for computational expediency, special consideration must also be taken to ensure efficiency of the overall scheme when the upper limit  $t$  is large and one of the matrices  $A$ ,  $C$  or  $D$  is unstable. Also one must economize memory requirements associated with high dimensionality of the augmented matrix when computing the matrix exponential. These considerations will be elaborated in Sections 3.6 and 3.7 where we give precise algorithms for the computation of the matrices  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$  respectively.

### 3.4 Numerical Method for the Matrix Exponential

Several numerical methods are available for the computation of the matrix exponential [22]. Among these, an approximation method based on Padé series is found to be satisfactory [23]. An important component in any numerical routine for matrix exponential is the scaling of the matrix argument prior to the series calculation. Due to the simple result that  $e^{At} = (e^{At/2})^2$ , a scale factor in terms of powers of two (i.e.  $2^m$ ) is often used. In this scheme, one can recover the actual value of the original matrix exponential by performing  $m$  squarings on the matrix exponential of the scaled matrix. The index  $m$  is determined based on the desired size for the scaled matrix. In our algorithm, scaling is applied to the original matrix until its  $\infty$ -norm  $\|A\|_\infty$  falls below  $1/2$ .

As mentioned above, the preferred series approximation in our computation of the matrix exponential is the Padé series. Let's review some of the unique features associated with the

Padé series for the case of a scalar function  $\mathcal{F}(z)$ . On its most basic terms, it is a rational function of  $z$  of a preselected order that approximates the function  $\mathcal{F}(z)$ . For a given choice of the order of the numerator (say  $N$ ) and of the denominator (say  $M$ ), the Taylor series representation of this Padé series must match the power series representation of  $\mathcal{F}(z)$  for the first  $(N + M + 1)$  terms. Namely,

$$\mathcal{F}(z) \sim P_M^N(z) = \frac{\sum_{i=0}^N A_i z^i}{\sum_{i=0}^M B_i z^i} \quad (3.10)$$

In fact, the most common form of the Padé series is known as the diagonal sequence where the numerator and the denominator have the same order (i.e.  $M = N$ ). While it is known that the Padé series for the matrix exponential  $\mathcal{F}(z) = e^z$  converges only slightly faster than the Taylor series for a scalar argument, the improvement becomes significant for a matrix argument. In the matrix case, Padé series involves computation of a numerator matrix  $\mathcal{N}(At)$  and of a denominator matrix  $\mathcal{D}(At)$ . For a diagonal Padé series of order  $N$ , we have

$$\begin{aligned} \mathcal{N}(At) = & \text{I} + \frac{(2N-1)! N!}{(2N)! (N-1)!} At + \frac{(2N-2)! N!}{(2N)! 2! (N-2)!} (At)^2 \\ & + \cdots + \frac{(2N-i)! N!}{(2N)! i! (N-i)!} (At)^i + \cdots + \frac{N!}{(2N)!} (At)^N \end{aligned}$$

and

$$\begin{aligned} \mathcal{D}(At) = & \text{I} - \frac{(2N-1)! N!}{(2N)! (N-1)!} At + \frac{(2N-2)! N!}{(2N)! 2! (N-2)!} (At)^2 \\ & - \cdots + (-1)^i \frac{(2N-i)! N!}{(2N)! i! (N-i)!} (At)^i + \cdots + (-1)^N \frac{2N!}{(2N)!} (At)^N \end{aligned}$$

The matrix exponential is simply given by

$$e^{At} = \mathcal{D}^{-1}(At) \mathcal{N}(At) \quad (3.11)$$

Invertibility of  $\mathcal{D}(At)$  is ensured by proper scaling of the matrix argument  $At$ .

Another important consideration in the Padé series is its length  $N$ . Assuming that the matrix  $At$  has been scaled such that  $\|At\|_\infty$  is less than  $1/2$ , the parameter  $N$  can be chosen according to [23] such that

$$2^{3-2N} \frac{(N!)^2}{(2N)! (2N+1)!} \leq \epsilon \quad (3.12)$$

where  $\epsilon$  is a given desired tolerance for accuracy.

With  $N$  determined in the above manner, the nominal error in a Padé series approximation can be thought of as the exact calculation of a matrix exponential for a “nearby” matrix  $(At + E)$  where  $E$  is the error matrix with  $\|E\|_\infty \leq \epsilon \|At\|_\infty$ . The relative error of the approximation is bounded by the following inequality,

$$\frac{\|e^{(At+E)} - e^{At}\|_\infty}{\|e^{At}\|_\infty} \leq \epsilon \|At\|_\infty e^{\epsilon \|At\|_\infty} \quad (3.13)$$

Thus, reducing the  $\infty$ -norm of the matrix  $At$  would indeed improve the numerical accuracy of the matrix exponential. It has also been shown that methods by series approximation

yield better accuracy if the matrix argument has been preconditioned [29]. Additional improvement may therefore be gained by first preconditioning the original matrix. Another immediate benefit of lowering the  $\infty$ -norm of the matrix being exponentiated is that the actual scaling factor  $m$  needed would also be smaller; thereby resulting in a fewer number of matrix multiplications in the squaring procedure. As usual, preconditioning a matrix tends to bring singular values of that matrix closer together (i.e. lower the condition number), thus avoiding situation where the scaling factor is predominantly determined by a few large singular values, and causing significant loss of precision related to the set of small singular values. The most common method used in the precondition of a matrix is Osborne's method [25], which minimizes the Frobenius norm of that matrix (and thus indirectly lowering its  $\infty$ -norm). However, extensive tests conducted so far seem to indicate that preconditioning of a matrix does not yield significant reduction in the  $\infty$ -norm and a smaller scaling factor. However, Osborne's method is relatively light in terms of computational burden. The procedure of preconditioning a matrix is nonetheless recommended from the point of view of improved accuracy ([26], [29]).

In the implementation of our design algorithm for optimal low-order controller synthesis [18], a value of  $\epsilon = 10^{-8}$  has been selected, requiring therefore a 4-term Padé series (i.e.,  $N = 4$ ) in the evaluation of the matrices  $\mathcal{X}^i(t)$ ,  $\mathcal{L}^i(t)$  and  $\mathcal{M}^i(t)$  of equations (2.58)-(2.60). Additional considerations in the implementation of the proposed method for computing  $\mathcal{X}(t)$  and  $\mathcal{M}(t)$  are given in Sections 3.6 and 3.7.

### 3.5 Preconditioning With Scaling and Rotation

We can liken the original cost function calculation method (that of an eigenvalue-eigenvector decomposition) as a method that does best when the eigenvalues are widely scattered (at least, distinct). The matrix series calculation excels when a matrix is homogenized. For the particular use in a controller optimization scheme, the matrices encountered usually have widely spaced eigenvalues, but degeneracies in these matrices may still exist.

Scaling the argument matrix to a Padé series to render its  $\infty$ -norm less than 1/2 accommodates a large magnitude eigenvalue (stable or unstable). This same scaling would lead to roundoff errors for a small magnitude eigenvalue in the same matrix. This potential problem renewed interest in finding some method that went beyond Osborne's method in homogenizing a matrix. We attempt such a method by combining the row and column scaling of Osborne's method with an inverse Jacobi rotation.

Suppose that one chooses a pair of columns and their corresponding rows in a matrix. Can one find a single rotation and two scaling terms to reduce the Frobenius norm of the matrix? Consider

$$\begin{aligned}
 A_{i+1} &= T_i^{-1} A_i T_i \\
 &= \begin{bmatrix} \vdots & & & & \\ \dots & \frac{\cos \theta}{\sigma_1} & \dots & -\frac{\sin \theta}{\sigma_2} & \dots \\ & \vdots & \ddots & \vdots & \\ \dots & \frac{\sin \theta}{\sigma_1} & \dots & \frac{\cos \theta}{\sigma_2} & \dots \\ & \vdots & & \vdots & \end{bmatrix} \begin{bmatrix} \vdots & & & & \\ \dots & A_{ii} & \dots & A_{ij} & \dots \\ & \vdots & \ddots & \vdots & \\ \dots & A_{ji} & \dots & A_{jj} & \dots \\ & \vdots & & \vdots & \end{bmatrix}
 \end{aligned}$$

$$\times \begin{bmatrix} \vdots & \vdots \\ \cdots & \sigma_1 \cos \theta & \cdots & \sigma_1 \sin \theta & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & -\sigma_2 \sin \theta & \cdots & \sigma_2 \cos \theta & \cdots \\ \vdots & \vdots \end{bmatrix}$$

we want to determine a rotation angle  $\theta$  and two scaling terms  $\sigma_1$  and  $\sigma_2$  such that

$$\min_{\theta, \sigma_1, \sigma_2} \|T_i^{-1} A_i T_i\|_F.$$

It would then be hoped that a succession of these transformations would lead  $A_n$  toward a minimum Frobenius norm value. The Frobenius norm was chosen to see if an easy means for determining  $\theta$ ,  $\sigma_1$ , and  $\sigma_2$  was even possible. Unfortunately, there is no easy means for finding these scalings or rotations. The amount of work needed is significant, and thus this approach was abandoned. It seemed that, among all the alternatives, Osborne's method was still the best.

### 3.6 Detailed Algorithm for Computing $\mathcal{X}(t)$

As seen in Section 3.3, the matrix  $\mathcal{X}(t)$  can be evaluated in terms of a matrix exponential as shown in equation (3.6). Conceptually, it is a simple and straightforward procedure to compute the matrix exponential of any arbitrary matrix using the Padé series discussed in Section 3.4. However, it becomes a nontrivial task when we try to implement an efficient algorithm that examines carefully the issues related to accuracy, speed and memory requirements. To understand the basic difficulties, consider in detail the components of the matrix exponential used according to our problem defined in equations (2.58) and (2.59) for the matrices  $\mathcal{X}^i(t_f)$  and  $\mathcal{L}^i(t_f)$

$$\exp \left\{ \begin{bmatrix} -A & B \\ 0 & C \end{bmatrix} t \right\} = \begin{bmatrix} e^{-At} & e^{-At} \mathcal{X}(t) \\ 0 & e^{Ct} \end{bmatrix} \quad (3.14)$$

where  $A = C^T = F^i + \alpha^i I$ . Clearly, if the system matrix  $A$  is stable (i.e. all the eigenvalues of the matrix  $A$  have negative real parts), one could easily encounter numerical overflow when evaluating the term  $e^{-At}$  even though the matrix integrals  $\mathcal{X}(t)$  and  $\mathcal{L}(t)$  are perfectly well-behaved. The overflow problem occurs most likely in the final squaring process. To arrive at a feasible approach in the evaluation of  $\mathcal{X}(t)$ , one needs to examine in detail the steps taken in arriving at the matrix exponential of the original matrix starting from that of a scaled matrix (i.e. in the squaring process).

Let's assume that one has scaled the input matrix  $A$  by  $A\Delta t$  where  $\Delta t$  is a reasonably small time interval given by  $\Delta t = t/n = t/2^m$ . Thus, we need to first evaluate

$$\exp \left\{ \begin{bmatrix} -A & B \\ 0 & C \end{bmatrix} \Delta t \right\} \text{ where } \Delta t = t/n = t/2^m.$$

For notation convenience, we define

$$\exp \left\{ \begin{bmatrix} -A & B \\ 0 & C \end{bmatrix} \Delta t \right\} = \begin{bmatrix} e^{-A\Delta t} & e^{-A\Delta t} \int_0^{\Delta t} e^{A\tau} B e^{C\tau} d\tau \\ 0 & e^{C\Delta t} \end{bmatrix} = \begin{bmatrix} D & E \\ 0 & F \end{bmatrix}. \quad (3.15)$$

Furthermore, let  $W = \exp(A\Delta t) = D^{-1}$ . Now we can write our result as follows,

$$\mathcal{X}(t) = W^n [D^{n-1}E + D^{n-2}EF + D^{n-3}EF^2 + \dots + EF^{n-1}], \quad (3.16)$$

or

$$\mathcal{X}(t) = W[E + WEF + W^2EF^2 + \dots + W^{n-1}EF^{n-1}]. \quad (3.17)$$

The above results are produced by performing  $m$  squarings of  $\begin{bmatrix} D & E \\ 0 & F \end{bmatrix}$  and taking the appropriate submatrix for  $\mathcal{X}(t)$ . In our application (cf. equations (2.58)-(2.59)), the solution would therefore involve products of matrices of size  $2(n+r+n')$ . Close examination of equation (3.17) leads to the following algorithm involving only product of matrices of size  $(n+r+n')$  with the final result achievable in  $m$  steps,

$$\begin{aligned} \text{Step 1: } P_1 &= W, & Q_1 &= E, & R_1 &= F \\ \text{Step 2: } P_2 &= P_1^2, & Q_2 &= Q_1 + P_1 Q_1 R_1, & R_2 &= R_1^2 \\ &= W^2, & &= E + WEF, & &= F^2 \\ &\dots & \dots &= \dots, & \dots &= \dots \\ \text{Step } m: P_m &= P_{m-1}^2, & Q_m &= Q_{m-1}, & R_m &= R_{m-1}^2 \\ &= W^n, & &+ P_{m-1} Q_{m-1} R_{m-1}, & &= F^n \end{aligned}$$

Finally,  $\mathcal{X}(t) = WQ_m$ . It should be noted that one can "absorb" this extra factor of  $W$  ( $= e^{A\Delta t}$ ) into the matrix  $Q_1$  without any change to the above algorithm (i.e. starting the above algorithm with  $Q_1 = WE$  instead). This removes the need to retain the matrix  $W$  throughout the computation.

Finally, one notes that the terms  $P_i$  or  $R_i$  for  $(i = 1, m)$  may underflow and become a null matrix for some  $i$ ; in particular when the scaling factor is large (i.e.  $m$  large). When this situation happens, one can simply truncate the series calculation for  $\mathcal{X}(t)$  up to the  $i^{th}$  step in the above algorithm since all of the significant (and nonzero) terms have already been accumulated into the matrix  $Q_i$ .

### 3.7 Detailed Algorithm for Computing $\mathcal{M}(t)$

Here the numerical algorithm is a bit involved compared to the one given for  $\mathcal{X}(t)$ . This is largely due to the increased complexity of the argument of the matrix exponential. Following the procedure described in Section 3.6, let's perform a scaling upon the input matrix  $A$  by  $A\Delta t$  such that computation of the matrix quantities  $\mathcal{M}_o, H, J, P, U$  and  $W = V^{-1}$  is well-behaved. These quantities are defined from the following matrix exponential,

$$\exp \left\{ \begin{bmatrix} A & Be^{Ct/2} & 0 \\ 0 & -C & e^{Ct/2}D \\ 0 & 0 & E \end{bmatrix} \Delta t \right\} = \begin{bmatrix} P & He^{Ct/2} & \mathcal{M}_o \\ 0 & V & e^{Ct/2}J \\ 0 & 0 & U \end{bmatrix} \quad (3.18)$$

Due to the possible numerical underflow in the matrix  $e^{Ct/2}$  for large  $t$ , the matrices  $H$  and  $J$  are computed directly from the following definitions,

$$H = \int_0^{\Delta t} e^{A\tau} B e^{C\tau} d\tau e^{-C\Delta t} \quad (3.19)$$

and

$$J = e^{-C\Delta t} \int_0^{\Delta t} e^{C\tau} D e^{E\tau} d\tau \quad (3.20)$$

However, the computation of  $\mathcal{M}_o$  in equation (3.18) can still underflow due to its explicit dependence on  $e^{Ct/2}$ . For the calculation of the matrix  $\mathcal{M}(t)$ , ideally it can be obtained from  $m$  squarings of equation (3.18). If carried out in this manner, potential numerical overflow is eminent since, according to our equation for  $\mathcal{M}^i(t_f)$  in (2.60), we have  $A^T = C = E^T = F^i + \alpha^i I$ . Hence, if the matrix  $C$  is stable, then the matrix exponential  $e^{-Ct} = V^n$  will become unbounded. To bypass this difficulty, as in the calculation for  $\mathcal{X}(t)$ , one needs to conduct the squaring algorithm explicitly. It can be shown that the matrix  $\mathcal{M}(t)$  can be computed as

$$\begin{aligned} \mathcal{M}(t) = & P^{n-1}\mathcal{M}_o + P^{n-2}\mathcal{M}_o U + P^{n-3}\mathcal{M}_o U^2 + \dots + P\mathcal{M}_o U^{n-2} + \mathcal{M}_o U^{n-1} \\ & + HW^2J + HW^3JU + PHW^3J + PHW^4JU + P^2HW^4J + \dots + P^{n-2}HW^nJ \end{aligned} \quad (3.21)$$

This formulation no longer involves the matrix  $V$ . The above series for  $\mathcal{M}$  can be decomposed into two parts—one that contains the matrix  $\mathcal{M}_o$  and the other that does not. The terms involving  $\mathcal{M}_o$  can be thought of as

$$[I \ 0] \begin{bmatrix} P & \mathcal{M}_o \\ 0 & U \end{bmatrix}^n \begin{bmatrix} 0 \\ I \end{bmatrix} \quad (3.22)$$

which can be performed by  $m$  squarings. The remaining terms involving  $H, J, W, P$ , and  $U$  are computationally intensive and are of the form

$$\sum_{i=0}^{n-2} \sum_{j=0}^{n-2} P^i HW^{2+i+j} JU^j \text{ where } 2+i+j \leq n. \quad (3.23)$$

Without the restriction that  $2+i+j \leq n$ , this would have been formed as the product of two easily computed series,

$$(H + PHW + P^2HW^2 + \dots)W^2(J + WJU + W^2JU^2 + \dots) \quad (3.24)$$

An efficient procedure for computing the final matrix  $\mathcal{M}(t)$  is to merge both the easily computed portion given in equation (3.22) and the more difficult series in equation (3.23) into a sequence of  $m$  steps:

<i>Step 0 :</i>					
$\mathcal{M}_o$	$H$	$J$	$P$	$U$	$W$
<i>Step 1 :</i>					
$\mathcal{M}_1 = P\mathcal{M}_o + \mathcal{M}_o U$ $+ HW^n J$	$H_1 = H$ $+ PHW$	$J_1 = J$ $+ WJU$	$P^2$	$U^2$	$W^2$
<i>Step 2 :</i>					
$\mathcal{M}_2 = P^2\mathcal{M}_1 + \mathcal{M}_1 U^2$ $+ H_1 W^{n-2} J_1$	$H_2 = H_1$ $+ P^2 H_1 W^2$	$J_2 = J_1$ $+ W^2 J_1 U^2$	$P^4$	$U^4$	$W^4$
<i>Step 3 :</i>					
$\mathcal{M}_3 = P^4\mathcal{M}_2 + \mathcal{M}_2 U^4$ $+ H_2 W^{n-6} J_2$	$H_3 = H_2$ $+ P^4 H_2 W^4$	$J_3 = J_2$ $+ W^4 J_2 U^4$	$P^8$	$U^8$	$W^8$
...	...	...	...	...	...
<i>Step j :</i>					
$\mathcal{M}_j = P^{2^{j-1}}\mathcal{M}_{j-1} + \mathcal{M}_{j-1} U^{2^{j-1}}$ $+ H_{j-1} W^{n+2-2^j} J_{j-1}$	$H_j = H_{j-1}$ $+ P^{2^{j-1}} H_{j-1} W^{2^{j-1}}$	$J_j = J_{j-1}$ $+ W^{2^{j-1}} J_{j-1} U^{2^{j-1}}$	$P^{2^j}$	$U^{2^j}$	$W^{2^j}$
...	...	...	...	...	...
<i>Step m :</i>					
$\mathcal{M}_m = P^{n/2}\mathcal{M}_{m-1} + \mathcal{M}_{m-1} U^{n/2}$ $+ H_{m-1} W^2 J_{m-1}$	$H_m = H_{m-1}$ $+ P^{n/2} H_{m-1} W^{n/2}$	$J_m = J_{m-1}$ $+ W^{n/2} J_{m-1} U^{n/2}$	$P^n$	$U^n$	$W^n$

where the final matrix  $\mathcal{M}(t) = \mathcal{M}_m + H_m W^2 J_m$ . Due to potential numerical underflow, the term  $W^{i-2}$  is not accurately obtained from the product  $W^i V^2$  where  $V = W^{-1}$ . Indeed, one needs to recompute the term  $W^{n+2-2^j}$  at each step of the above algorithm. This could become the major drawback in our scheme even though we have used an efficient matrix exponentiation routine that computes  $W^i$  requiring at most  $2\log_2(i)$  matrix multiplies.

Further simplification of the above algorithm can be achieved if we make use of the fact that we have  $A = E = C^T$  (cf. equation (2.60)) and therefore  $U = P = W^T$ . If in addition  $W^n$  is zero (or effectively so), restriction on the indices  $i$  and  $j$  of  $2 + i + j \leq n$  in equation (3.23) becomes inconsequential. Hence we can express

$$\sum_{i=0}^{n-2} \sum_{j=0}^{n-2} P^i H W^{2+i+j} J U^j = (H + PHW + P^2 HW^2 + \dots) W^2 (J + WJU + W^2 JU^2 + \dots)$$

resulting in a simpler algorithm that involves the following three series,

$$\begin{aligned} (a) & [I \ 0] \begin{bmatrix} P & \mathcal{M}_o \\ 0 & U \end{bmatrix}^n \begin{bmatrix} 0 \\ I \end{bmatrix} \\ (b) & (H + PHW + P^2 HW^2 + \dots) \\ (c) & (J + WJU + W^2 JU^2 + \dots) \end{aligned}$$

This algorithm can again be computed in  $m$  steps as follows,

$$\begin{array}{lll}
\text{Step 0 :} & & \\
\mathcal{M}_0 & H & J \\
\text{Step 1 :} & & \\
\mathcal{M}_1 = P\mathcal{M}_0 + \mathcal{M}_0U & H_1 = H + PHW & J_1 = J + WJU \\
\text{Step 2 :} & & \\
\mathcal{M}_2 = P^2\mathcal{M}_1 + \mathcal{M}_1U^2 & H_2 = H_1 + P^2H_1W^2 & J_2 = J_1 + W^2J_1U^2 \\
\text{Step 3 :} & & \\
\mathcal{M}_3 = P^4\mathcal{M}_2 + \mathcal{M}_2U^4 & H_3 = H_2 + P^4H_2W^4 & J_3 = J_2 + W^4J_2U^4 \\
\cdots & \cdots & \cdots \\
\text{Step } j : & & \\
\mathcal{M}_j = P^{2^{j-1}}\mathcal{M}_{j-1} & H_j = H_{j-1} & J_j = J_{j-1} \\
+ \mathcal{M}_{j-1}U^{2^{j-1}} & + P^{2^{j-1}}H_{j-1}W^{2^{j-1}} & + W^{2^{j-1}}J_{j-1}U^{2^{j-1}} \\
\cdots & \cdots & \cdots \\
\text{Step } m : & & \\
\mathcal{M}_m = P^{n/2}\mathcal{M}_{m-1} & H_m = H_{m-1} & J_m = J_{m-1} \\
+ \mathcal{M}_{m-1}U^{n/2} & + P^{n/2}H_{m-1}W^{n/2} & + W^{n/2}J_{m-1}U^{n/2}
\end{array}$$

If, for some index  $j \neq m$ ,  $W^i$  (and likewise  $U^i$  and  $P^i$ ) is zero or nearly so, then the calculation of  $\mathcal{M}(t)$  is reduced to  $\mathcal{M}(t) = H_j W^2 J_j$  since  $M_j = 0$  where  $i = 2^j$ .

### 3.8 Basic Test Results

A direct evaluation of  $\mathcal{X}$  and  $\mathcal{M}$  on a degenerate system matrix provides a clear understanding of the severity of the errors in the diagonalization method. Consider

$$\begin{aligned}
F &= \begin{bmatrix} -0.5 & 0.25 & 0.333 \\ 1.0 & -1.5 & 0.0 \\ -1.5 & 0.75 & -1.0 \end{bmatrix} \\
\Gamma &= \begin{bmatrix} 1.0 & 7.0 & 4.0 \\ 3.0 & 9.0 & 6.0 \\ 5.0 & 2.0 & 8.0 \end{bmatrix} \\
H_c'^T Q H_c' + C'^T R C' &= \begin{bmatrix} 69.0 & 17.0 & 41.0 \\ 17.0 & 51.0 & 18.0 \\ 41.0 & 18.0 & 74.0 \end{bmatrix},
\end{aligned}$$

where  $F$  is constructed under a similarity transformation from a core matrix  $\Lambda$  and a transformation  $T$ ,

$$\Lambda = \begin{bmatrix} -1.0 & 1.0 & 0.0 \\ 0.0 & -1.0 & 1.0 \\ 0.0 & 0.0 & -1.0 \end{bmatrix}, \quad T = \begin{bmatrix} 1.0 & 1.0 & -1.0 \\ 2.0 & -2.0 & 2.0 \\ -3.0 & 3.0 & 3.0 \end{bmatrix},$$

Namely,  $F = T\Lambda T^{-1}$ . In the  $\mathcal{X}$  computation, the results were

$$\mathcal{X}_{diag} = \begin{bmatrix} 58.37 & 58.97 & 30.58 \\ 58.97 & 57.83 & 45.91 \\ 30.58 & 45.90 & 00.00 \end{bmatrix}, \quad \mathcal{X}_{robust} = \begin{bmatrix} 61.90 & 66.04 & 19.98 \\ 66.04 & 71.96 & 24.70 \\ 19.98 & 24.70 & 31.80 \end{bmatrix}.$$

As one can see, there is really no correspondence between the two results. Furthermore, in comparing the results for  $\mathcal{M}$ , we have

$$\mathcal{M}_{diag} = \begin{bmatrix} -3.623 \times 10^9 & 4.997 \times 10^9 & -7.495 \times 10^9 \\ 3.081 \times 10^{17} & 6.162 \times 10^{17} & -9.244 \times 10^{17} \\ 2.054 \times 10^{17} & 4.108 \times 10^{17} & -6.162 \times 10^{17} \end{bmatrix},$$

$$\mathcal{M}_{robust} = \begin{bmatrix} 2211. & 2519. & 1230. \\ 1860. & 2146. & 1153. \\ 1817. & 2133. & 1475. \end{bmatrix}.$$

With these large discrepancies, it is clear that design optimization based on the diagonalization method would fail when defective eigenvalues appear in the closed-loop system.

### 3.9 A Two-Mass-Spring Design Problem

In this model, the open-loop system has defective eigenvalues at the origin; thus success of the diagonalization-based approach depends on the removal of this degeneracy through feedback. Hence, selection of the initial controller guess plays a key role. We consider two setups to illustrate the problems encountered by the algorithm based on diagonalization.

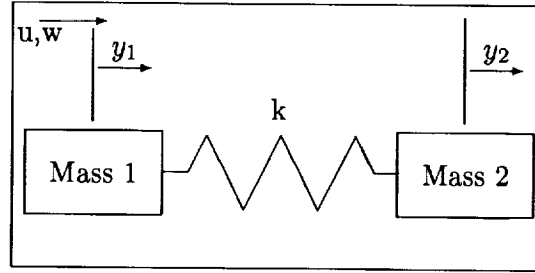


Figure 3.1: A Two-Mass-Spring Mass System

Equations for the dynamic model are given below.

$$\begin{aligned} m_1 \ddot{y}_1 &= k(y_2 - y_1) + u + w \\ m_2 \ddot{y}_2 &= k(y_1 - y_2) \end{aligned} \quad (3.25)$$

or

$$\frac{d}{dt} \begin{bmatrix} y_1 \\ y_1' \\ y_2 \\ y_2' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -k/m_1 & 0 & k/m_1 \\ 0 & 0 & 1 & 0 \\ 0 & k/m_2 & 0 & -k/m_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_1' \\ y_2 \\ y_2' \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m_1 \\ 0 \\ 0 \end{bmatrix} (u + w) \quad (3.26)$$

where  $m_1 = m_2 = k_1 = k_2 = 1$ . The problem is to control the displacement of the second mass by applying a force to the first mass as shown in Figure 3.1. At the start, it is simple to verify that the basic open-loop system has a pair of defective eigenvalues at the origin.

We have chosen a second-order controllable canonical form for our controller model,

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ A_{21} & A_{22} \end{bmatrix}; & B &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ C &= [C_{11} \ C_{12}]; & D &= [D_{11}] \end{aligned}$$

The control design problem is to minimize the  $H^2$ -norm of the following closed-loop transfer function  $T_{y_2w}$  between the disturbance  $w$  and the displacement  $y_2$  of the second mass using the controller design parameters  $A_{21}$ ,  $A_{22}$ ,  $C_{11}$ ,  $C_{12}$  and  $D_{11}$ . From the initial design guesses,

$$A_{21} = -2, A_{22} = -1, C_{11} = 0, C_{12} = 0.5 \text{ and } D_{11} = 0$$

The original method based on diagonalization in *SANDY* [18] converges to the optimal design gains,

$$A_{21} = -0.8571, A_{22} = -0.9258, C_{11} = 0, C_{12} = -0.4535 \text{ and } D_{11} = -0.2449$$

and the optimum value of  $\|T_{y_2w}\|_2^2 = 7.71838215122$ . A summary of the resulting closed-loop eigenvalues is given in Table 3.1.

Table 3.1: Closed-Loop Eigenvalues of the Two-Mass-Spring System

Eigenvalue			Damping	Freq (Hz)
-0.2290	$\pm$	$0.3397i$	0.559	0.0652
-0.1553	$\pm$	$0.8480i$	0.180	0.1372
-0.0786	$\pm$	$1.2950i$	0.061	0.2065

One cannot initiate the search using a compensator design of zero gains (i.e.,  $A_{21} = A_{22} = C_{11} = C_{12} = D_{11} = 0$ ) because, in this case, the closed-loop system would have two pairs of degenerate eigenvalues at the origin; one for the rigid-body mode and the other from the open-loop compensator poles. The early algorithm based on diagonalization recognizes that Jordan blocks are present and the numerical search is halted. Detection of a Jordan block within *SANDY* is done by looking at the condition of the eigenvector matrix. If this condition goes above  $10^{12}$  or its inverse is less than  $10^{-12}$ , the design search is terminated with a failed status. Condition of the eigenvector matrix influences the accuracy of the matrices  $\mathcal{X}$  and  $\mathcal{M}$ . At some point, the loss in accuracy will affect the overall optimization process. To alleviate this problem, it was suggested that one simply re-start the design algorithm with any compensator design (stabilizing or not) that initially produces a non-degenerate closed-loop system. This approach does not provide a satisfactory solution in general.

A more insidious possibility is exemplified by the following initial design guess,

$$A_{21} = -2, A_{22} = -1, C_{11} = 0, C_{12} = 0 \text{ and } D_{11} = 0$$

It turns out that this case of Jordan blocks is not detected from the condition of the eigenvector matrix. The diagonalization-based algorithm fluttered around the initial point and finally terminated with no improvement to the original design guess. The condition number of the eigenvector matrix associated with this problem was  $2.302 \times 10^6$ , which would generally be considered acceptable.

The robust form of the algorithm allows convergence of the controller gains in *all* the three initial guesses shown. The final result is the same in each case and equal to

$$A_{21} = -0.8571, A_{22} = -0.9258, C_{11} = 0, C_{12} = -0.4535 \text{ and } D_{11} = -0.2449.$$

The result is the same as the one determined from a successful run using the diagonalization algorithm. A history of the inverse of the condition number of the eigenvector matrix for the initial design guess of

$$A_{21} = -2, A_{22} = -1, C_{11} = 0, C_{12} = 0.5, \text{ and } D_{11} = 0,$$

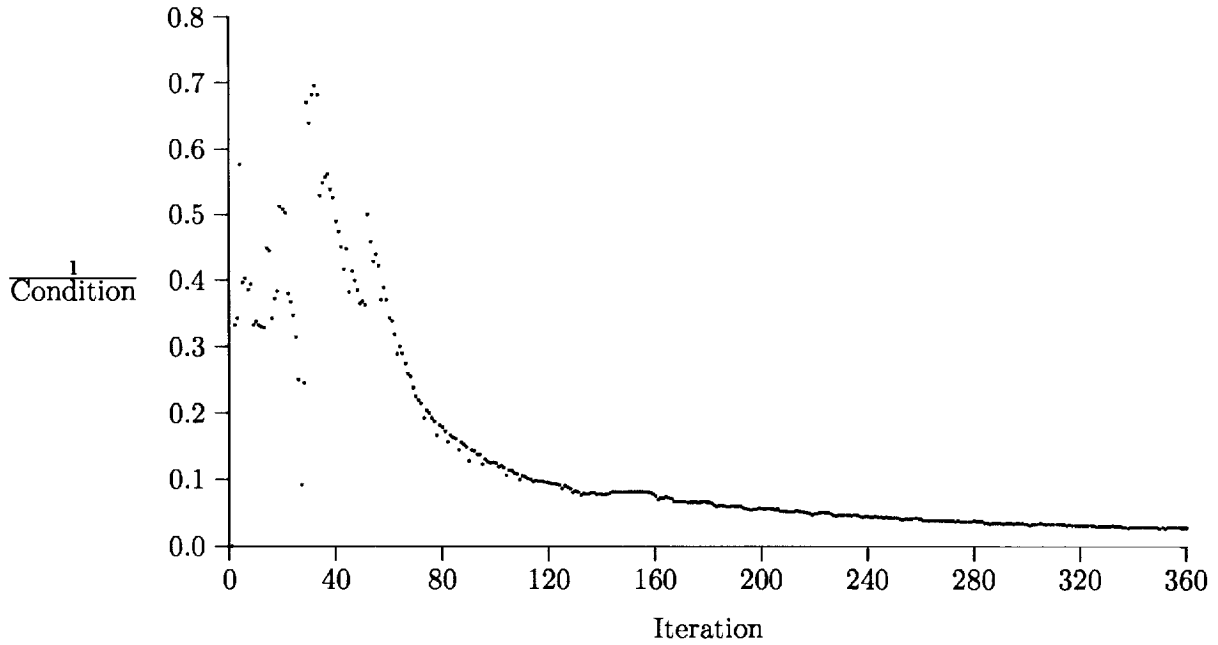


Figure 3.2: Behavior of Condition Number over the Entire Design Optimization

is shown in Figure 3.2. One can see that condition of the eigenvector matrix varies significantly over the entire run. Conceivably, the inverse condition number may jump back to a low value in a typical design iteration. Not very visible in this plot is the value at the first iteration of  $1.8 \times 10^{-8}$  for the chosen initial guesses. After the first iteration, the inverse condition number jumps to 0.166.

The main difference between the robust (Pade-series-based) form and the diagonalized form is in the CPU time of the overall computation. Results are obtained for a VAX/VMS-Workstation DEC-3500 as follows: CPU time of 19.59 seconds with the algorithm based on diagonalization, and 97.36 seconds using the proposed robust algorithm. The increase in computational load is expected and constitutes the basic trade-off between reliability and speed. The proposed algorithm is more reliable and with this advantage does take somewhat longer in computational time.

Although a simple fix to the diagonalized form is to start with a non-degenerate closed-loop system by using a different initial compensator design, it has been found that the algorithm could occasionally break down due to the presence of near degeneracies. Thus, for a reliable design method, the solution algorithm must treat degeneracies as a common occurrence. This situation is more evident in the optimal output-feedback control design for high-order structural models with closely packed flexible modes, and in the design procedure of closed-loop transfer recovery. These cases include a situation where two nominally independent modes appear as a Jordan block due to roundoff error in a higher order system (usually 30<sup>th</sup>-order or higher).

For efficiency, the numerical algorithm currently implemented combines the robust algorithm with the faster diagonal form through a "switch" based on a threshold of  $2 \times 10^{-5}$  in the inverse condition of the eigenvector matrix. This value is found to be adequate in detecting defective degenerate cases and computational accuracy of the run is improved by the intermittent calls to the robust form.

### 3.10 Degeneracy During Optimization

Presented here is a model reduction problem that develops a Jordan block *during* the optimization process [31]. Consider a model with closely spaced modes,

$$F = \begin{bmatrix} -1 & 0 & 0 \\ 0.0005 & -1.000001 & 0 \\ 0.0005 & 0.0005 & -1.00001 \end{bmatrix}, \Gamma = \begin{bmatrix} 1.1 \\ 1.2 \\ 1.3 \end{bmatrix}, H = [2.1 \ 2.2 \ 2.3], D = [0]$$

Parameter optimization is used to reduce this model to a second-order system by minimizing the  $H^2$ -norm of the error. Starting with the following arbitrarily chosen guess

$$A_c = \begin{bmatrix} -0.9995 & 1 \\ 2 & -1 \end{bmatrix}, B_c = \begin{bmatrix} -1.414 \\ 1.414 \end{bmatrix}, C_c = [-5.616 \ 0], D_c = [0]$$

The optimization converges to the following reduced-order model,

$$A = \begin{bmatrix} -1.027 & 0.012 \\ 1.436 & -1.625 \end{bmatrix}, B = \begin{bmatrix} -1.440 \\ 1.542 \end{bmatrix}, C = [-5.627 \ -0.107], D = [0]$$

with a final cost function of  $4.76 \times 10^{-7}$ . Diagnostics associated with this run indicate that while the gains were “optimized”, the solution is far from the optimal solution. Inverse condition number of the eigenvector matrix is  $1.26 \times 10^{-5}$  indicating the presence of degenerate modes.

This degeneracy is confirmed when the optimization was run using the robust form for the gradient computation. The optimized cost function now has a value of  $2.38 \times 10^{-12}$  which is much smaller than that achieved under the diagonalization algorithm. In addition, inverse condition number of the eigenvector matrix is now  $9 \times 10^{-6}$  indicating that degeneracies occur through the design optimization.

### 3.11 Conclusions

Numerical algorithms for computing matrix exponentials and integrals of matrix exponentials have been developed to handle cases where the system matrix has defective eigenvalues. Special formulation of these algorithms enables reliable and efficient computation of  $\mathcal{X}$  and  $\mathcal{M}$ . These algorithms have been incorporated into the computer-aided design package *SANDY* for synthesizing optimal output-feedback controllers. Numerical optimization combined with the given algorithms in the evaluation of the cost function and its gradients with respect to the controller design parameters is shown to have well-behaved convergence even when the closed-loop system becomes degenerate. Reliability of the algorithm is further demonstrated using typical design problems encountered in control of flexible structures. Clearly, this algorithm, when combined with a previous one based on diagonalization, would enhance significantly the overall reliability of optimal control synthesis using parameter optimization, thereby providing an effective automated design environment for multivariable control synthesis.

To further improve the efficiency of the robust algorithm, a hybrid form combining the benefits of both the diagonalization-based form and the robust form has been developed. There are two major components to this algorithm. The first one is a means of reliably decomposing the system matrix into two parts: a diagonal block containing nondefective

modes and diagonal subblocks containing the defective portions. Details are presented in Appendix A. The actual calculation of  $X$  and  $M$  are performed in the second step, consisting of elements in the diagonalization-based form, the robust form, and new cross-term calculations. Details are discussed in Appendix B.

## Chapter 4

# Control Design for a Large Space Structure

### 4.1 Introduction

In this chapter, we present a control design problem that poses significant difficulties to the early design algorithm based on diagonalization. The system model contains numerous densely packed flexible modes primarily due to the symmetry in the configuration of the structural design. There is a high likelihood of numerical degeneracies due to round-off in a finite-precision machine. The associated eigenvector matrix tends to be ill-conditioned and this problem occurs frequently during the design optimization.

### 4.2 JPL Large Space Structure

Defective degeneracy in low-order systems can be easily identified and the problem possibly remedied through simple changes in the model parameters. For example, transverse Dryden turbulence filters contain a pair of degenerate real roots; these filter poles can be perturbed slightly to produce a non-defective system with virtually no loss of accuracy in the calculation of power spectral densities. Most linear time-invariant controller state matrices can be represented in a canonical form with 2x2 blocks along its diagonal of the form

$$\begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix}. \quad (4.1)$$

Clearly a choice of  $\zeta = 1$  would lead to a degenerate subblock.

In control problems for large flexible mechanical systems such as space structures, causes of eigenvalue degeneracies are usually more subtle in nature than the simple case presented in Section 3.9 for a two-mass-spring system. The JPL large space structure has been carefully designed to simulate a lightweight, non-rigid and lightly damped structure in a weightless environment [28]. The structure itself resembles a large antenna with a central boom-dish apparatus and an extended dish consisted of hoop wires and 12 ribs. There are two torque actuators (labelled *HA1* and *HA10*) on the boom and dish structure to control the two angular degrees of freedom in pointing maneuvers, and force actuators at four rib root locations (labelled *RA1*, *RA4*, *RA7* and *RA10*) for vibration control. From the point of view of control design, it is a challenging problem, since the plant has many closely spaced modes and is of reasonably high order. There are a total of 30 modes in the basic structural

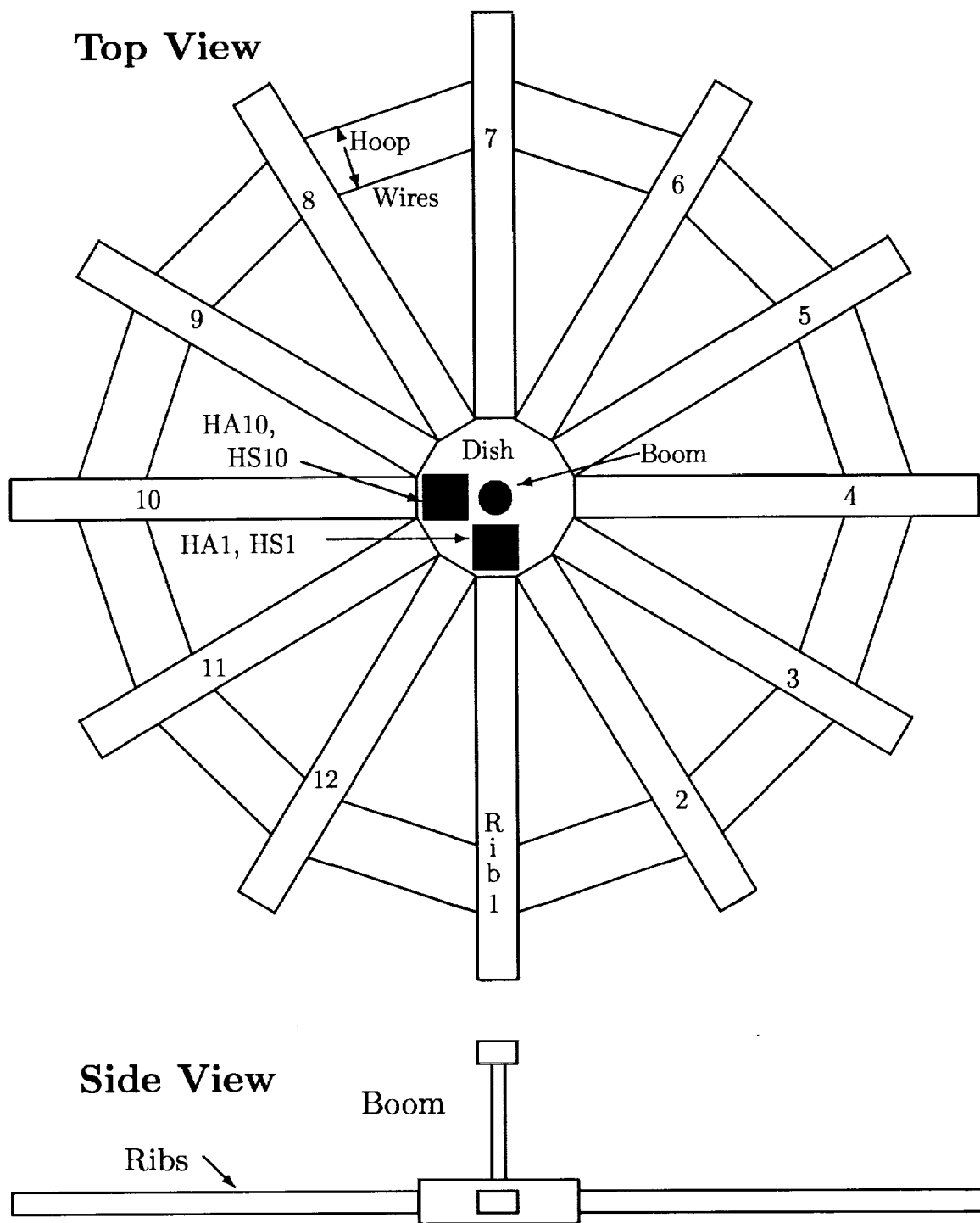


Figure 4.1: Antenna Structure

model. The flexible modes are lightly damped with damping ratios ranging from 0.007 to 0.01. The two rigid-body modes have a damping ratio of 0.12. Our design concept is to use two available angular displacement sensors *HS1* and *HS10* of the boom-dish apparatus and the two torquers *HA1* and *HA10* collocated with these sensors for control synthesis. With

Table 4.1: Open Loop Modes of the Antenna Structure

Eigenvalue	Damping	Freq (Hz)
-0.09500 ± 0.7860i	0.120	0.12600
-0.08575 ± 0.7093i	0.120	0.13704
-0.02802 ± 4.0024i	0.007	0.63701
-0.02929 ± 4.1844i	0.007	0.66598
-0.07405 ± 10.583i	0.007	1.68434
-0.07405 ± 10.583i	0.007	1.68434
-0.11310 ± 10.616i	0.007	2.57123
-0.11785 ± 16.384i	0.007	2.67929
-0.21365 ± 30.520i	0.007	4.85749
-0.21365 ± 30.520i	0.007	4.85749

this selection, 20 of the flexible modes associated primarily with the rib motion become uncontrollable and unobservable. These modes are removed by modal truncation from our plant synthesis model. Eigenvalues of the remaining 10 modes are shown in Table 4.1.

### 4.3 Controller Design

An optimal low-order controller is designed to dampen vibration of the antenna in response to external excitations. To evaluate the effectiveness of the control system, we perform the following test. The entire structure is agitated using the two boom-dish actuators for the first 6.4 seconds with an applied torque in the form of a square wave of 0.8 second in width and with an amplitude of 1 N-m. The control system is then activated right after the excitation has been removed, and responses of the excited structure at the sensors are examined. The design objective is to damp out the induced vibration as fast as possible without excessive use of controls. Note that the natural responses of the structure will take about a few minutes to decay to zero (Figures 4.2 and 4.3).

For practical implementation, the controller design is chosen to be of 6<sup>th</sup> order. The controller structure consists of a pair of tightly coupled second order systems each with dynamics of the form  $\begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\zeta\omega \end{bmatrix}$  a pair of actuator lag states,

$$\begin{aligned}
 A &= \begin{bmatrix} -50 & 0 & A_{13} & A_{14} & A_{15} & A_{16} \\ 0 & -50 & A_{23} & A_{24} & A_{25} & A_{26} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & A_{43} & A_{44} & 0 & 0 \\ 0 & 0 & A_{53} & A_{54} & 0 & 1 \\ 0 & 0 & A_{63} & A_{64} & A_{65} & A_{66} \end{bmatrix}, & B &= \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \\ B_{41} & B_{42} \\ B_{51} & B_{52} \\ B_{61} & B_{62} \end{bmatrix} \\
 C &= \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \end{bmatrix}, & D &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.
 \end{aligned} \tag{4.2}$$

The two first-order lag states in the controller model serve as roll-off filters, limiting the control bandwidth to less than 50 rad/sec. In the design optimization, we have a total of 28 design variables: 16 in the controller  $A$  matrix and 12 in the  $B$  matrix. The objective function for design optimization consists of a sum of weighted  $H^2$ -norms of physical response variables observed at different locations of the structure. It is of the form

$$J(t_f) = \lim_{t_f \rightarrow \infty} \frac{1}{2} \left\{ \sum_{i=1}^{12} Q_i E_\alpha [y_i^2(t_f)] + \sum_{j=1}^2 R_j E_\alpha [u_j^2(t_f)] \right\} \quad (4.3)$$

Note that the expectation operator  $E_\alpha[*]$  is for a system destabilized by a factor  $\alpha$ . Table 4.2 lists the design variables  $y_i$  and their corresponding penalty weightings  $Q_i$ . Also given in

Table 4.2: Design Variables: JPL Antenna Structure

Variable	$Q_i$	Description
$RS1$	4100	Rib #1 root velocity
$RS4$	3950	Rib #4 root velocity
$RS7$	3975	Rib #7 root velocity
$RS10$	4050	Rib #10 root velocity
$HS1$	16500	Hub angular velocity
$HS10$	15600	Hub angular velocity
$RS1$	1100	Rib #1 root displacement
$RS4$	1050	Rib #4 root displacement
$RS7$	1150	Rib #7 root displacement
$RS10$	1025	Rib #10 root displacement
$HS1$	3900	Hub angular displacement
$HS10$	4100	Hub angular displacement
Variable	$R_i$	Description
$HA1$	41	Hub torque actuator
$HA10$	40	Hub torque actuator

the table are the control design weightings  $R_j$  for the actuators  $HA1$  and  $HA10$ . Responses in the above objective function are evaluated to random disturbances of unit white-noise spectra applied simultaneously at all the hub and rib actuators. One may notice that the values of related weighting terms are perturbed about a nominal. With this nominal value as the weighting factor, the perturbation was for avoiding degenerate modes in the optimum design.

The design optimization begins with the following *arbitrary* initial guess on the controller matrices  $A$  and  $B$ ,

$$A = \begin{bmatrix} -50 & 0 & 1 & 0 & 0 & 0 \\ 0 & -50 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -4 & -4 \end{bmatrix}; \quad B = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

A destabilization factor  $\alpha$  of 0.071 was used to ensure that all the closed-loop eigenvalues have a real part less than  $-0.071$ . The optimization fails to converge when a destabilization

factor of greater than 0.075 was selected. This difficulty seems to be in moving the modes at 1.68Hz under this controller configuration, implying that additional degrees of freedom must be added to the controller structure given in equation (4.2), in order to improve the performance further.

While the optimization convergence itself took 13.5 hours on a VAX/VMS Workstation DEC-3500, the proposed algorithm for the calculation of the objective function and its gradients with respect to the design parameters is robust and leads to well-behaved design convergence. The final optimal values of the  $A$  and  $B$  matrices are

$$A = \begin{bmatrix} -50 & 0 & 2.874 & -2.270 & 1.7322 \times 10^{-3} & -2.2131 \times 10^{-4} \\ 0 & -50 & 1.225 & 0.7825 & 6.551 & -1.037 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -15.73 & -0.8799 & 0 & 0 \\ 0 & 0 & 1.560 & 0.2256 & 0 & 1 \\ 0 & 0 & 2.400 & -1.269 & -13.62 & -0.9810 \end{bmatrix}$$

$$B = \begin{bmatrix} 5.343 & -1.2310 \times 10^{-4} \\ 6.2118 \times 10^{-4} & 4.783 \\ 2.701 & -8.1595 \times 10^{-4} \\ 2.221 & 9.3152 \times 10^{-4} \\ -0.5147 & 5.379 \\ 1.614 & -1.252 \end{bmatrix} \quad (4.4)$$

Closed-loop responses of the sensor and control variables corresponding to this design are

Table 4.3: Boom-Dish-Controller Closed Loop Modes

Eigenvalue	Damping	Freq (Hz)
-0.086899 ± 0.6588i	0.1308	0.1058
-0.089071 ± 0.7410i	0.1193	0.1188
-0.3165 ± 3.624i	0.0870	0.5790
-0.2528 ± 3.790i	0.0666	0.6045
-0.2162 ± 4.112i	0.0525	0.6553
-0.2056 ± 4.185i	0.0491	0.6669
-0.074193 ± 10.58i	0.0070	1.684
-0.074589 ± 10.58i	0.0070	1.684
-0.1168 ± 16.15i	0.0072	2.570
-0.1253 ± 16.83i	0.0074	2.678
-0.2142 ± 30.52i	0.0070	4.857
-0.2143 ± 30.52i	0.0070	4.857
-49.99	1.000	7.956
-49.99	1.000	7.956

shown in Figures 4.2 and 4.3. The controlled responses decay to zero in about 20 sec after the excitation has been removed. Notice that the control torques are within the desired limits of 1 N-m; the results are obtained through the control design weights  $R_j$  in Table 4.2. This design example demonstrates the usefulness of a design algorithm for robust low-order controllers using parameter optimization, and the accompanying improvement of solution reliability using the algorithms described in Sections 3.6 and 3.7 for degenerate systems.

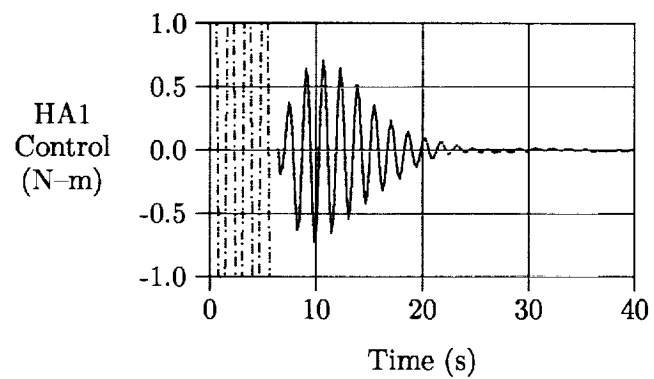
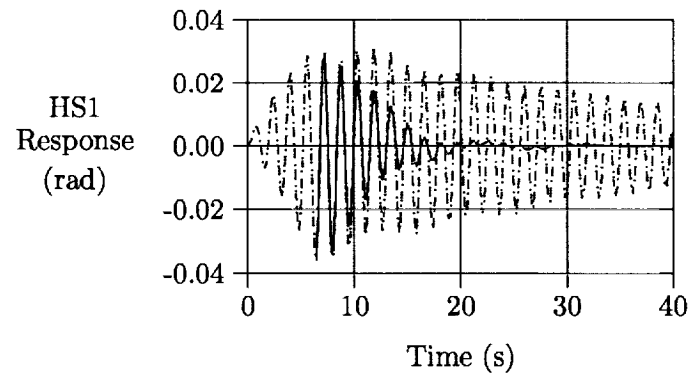


Figure 4.2: Hub Responses to Structure Excitation

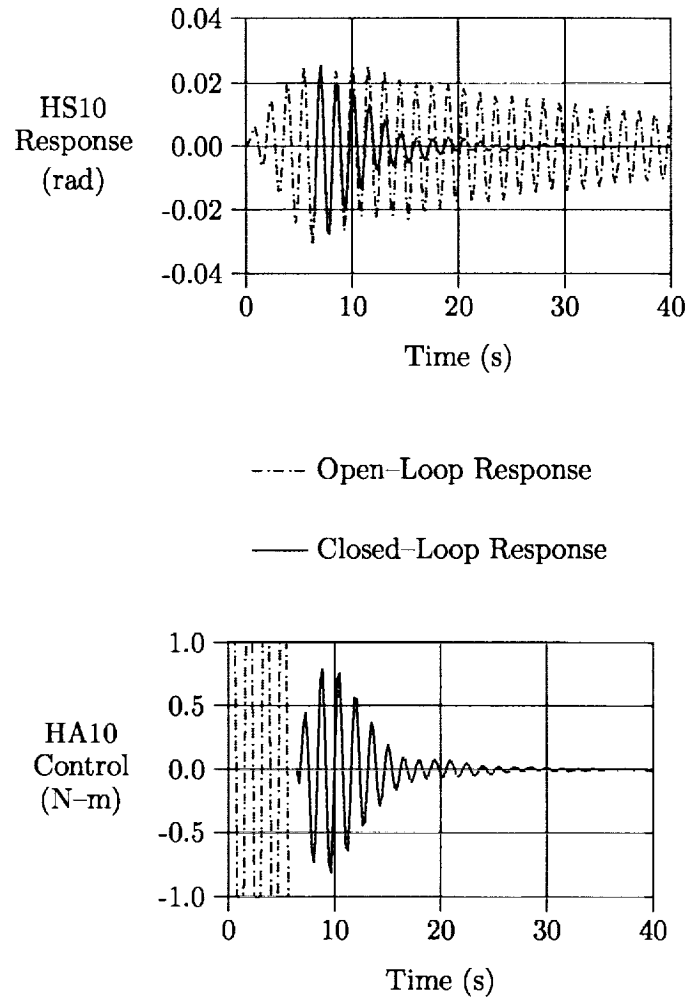


Figure 4.3: Hub Responses to Structure Excitation (cont.)



## Chapter 5

# Closed-Loop Transfer Recovery

### 5.1 Introduction

Multivariable control has evolved over the last decade. Methods for synthesizing control laws using optimal control from linear quadratic regulator to linear quadratic gaussian designs have been studied extensively. Robustness issues of LQG have led to the development of the concept of loop transfer recovery [1]. Recent work by Saberi et. al has fundamentally examined the loop transfer recovery based on a system theoretic approach [11]. In this chapter, we review briefly the concept of closed-loop transfer recovery which is an extension of the loop transfer recovery method for the recovery of closed-loop input/output behaviour. Observer-based controller designs for CLTR will be presented to illustrate the design concept. A new alternate approach based on numerical optimization is developed to solve CLTR designs for low-order compensators. This approach is then applied to the synthesis of a high-bandwidth control system for a rotorcraft in Chapter 7.

### 5.2 Analysis of Closed-Loop Transfer Recovery

For performance and robustness, multivariable control synthesis usually begins with the design of a state-feedback controller. A state-feedback design based on LQR techniques involves only the solution of Riccati equations and provides a convenient and efficient way to examine the control problems in terms of achievable performance and robustness. However, state-feedback design is not practical due to its requirement of noise-free measurement of all the states, but such a design can provide a target response for output-feedback designs using loop transfer recovery (LTR). Recovery of closed-loop transfer functions achieved under state feedback is classified under the category of closed-loop transfer recovery. This concept has been developed in [11]-[12] using full and reduced-order Luenberger observers. Designs can be solved using Riccati equations. This approach is later extended to arbitrary low-order compensators using numerical optimization.

The premise of CLTR is simple. Consider a linear time-invariant plant

$$\begin{aligned} \dot{x}(t) &= Fx(t) + \Gamma w(t) + Gu(t) & (\text{Dynamics}) \\ z(t) &= H_c x(t) + D_{cw} w(t) + D_{cu} u(t) & (\text{Criteria}) \\ y(t) &= H_s x(t) + D_{sw} w(t) + D_{su} u(t) & (\text{Sensors}) \end{aligned}$$

where  $w(t)$  represents the command/disturbance input vector and  $u(t)$  is the control input vector. We assume that a state-feedback design  $u = Kx$  has been obtained that achieves

the desired closed-loop properties in the transfer function  $T_{zw}(s)$  between the input  $w(t)$  and the output  $z(t)$ . Many different design methods can be used, e.g.  $H^2$ -optimal control [15],  $H^\infty$ -optimization [4], and others [33]. A state-feedback design is totally characterized by the gain matrix  $K$  and delineates the achievable performance. The actual performance is often limited due to a restricted set of measurements and the associated sensor noises. The problem reduces to finding output-feedback controllers to recover as much as possible the performance achieved under state feedback. The solution can be found using the concept of closed-loop transfer recovery with the recovery error defined by

$$E(s) = T_{zw}^o(s) - T_{zw}(s) = T_{zu}(s)M(s) \quad (5.1)$$

where  $T_{zw}(s)$  and  $T_{zw}^o(s)$  are the closed-loop transfer functions between the disturbance input  $w(t)$  and the criterion output  $z(t)$  of a state-feedback and observer-based output-feedback designs respectively. The transfer function  $T_{zu}(s)$  is simply the closed-loop transfer function between the control input  $u(t)$  and the criterion output  $z(t)$  under state feedback (independent of the output-feedback design). The transfer function  $M(s)$  takes on different forms depending on whether a full or a reduced-order Luenberger observer has been used [11]. The structure of the auxiliary transfer function  $M(s)$  is defined explicitly in Sections 5.5 and 5.6. The CLTR procedure for observer-based controllers addresses the minimization of the norm of  $\|M(s)\|$  whose solutions can be found from Riccati-based methods in either  $H^2$ - or  $H^\infty$ -based optimization. However, minimization of the norm of the actual error function  $\|E(s)\|$  can only be done using numerical optimization schemes to be discussed in Section 6.2.

### 5.3 The Special Coordinate Basis

Analysis of closed-loop transfer recovery can be effectively performed when the system model is transformed into a Special Coordinate Basis (SCB) [10]. Under the SCB transformation, many system properties can be identified: right and left invertibility, system invariant zeros, infinite zero order structure and the related geometric subspaces.

Appendix C presents an alternative approach underlying the conceptual development of the SCB transformation following similar notation as given in [10]. Other useful applications of the SCB properties are found in  $H^2$  and  $H^\infty$  optimization [33].

Let's consider for simplicity a strictly proper system. The basic notion of the S.C.B. is to divide up the system states into internal states and output states. The output states are either themselves outputs or are inputs to a cascade of integrators leading to the outputs. The state at the top of each cascade will either have one control term entering into its state equation, or it will not have any control input at all. Thus each cascade of output states will belong to one of two classes. The internal states are the remaining system states that do not fall into this category. The internal states can be collectively thought of as providing feedback to the system from the outputs (Figure 5.1). The class of internal states is subdivided into those states that are controllable with those control inputs not present in the dynamics of the output states, and those states which are not controllable with these inputs. This partition is slightly analogous to that of the output states. Note that if an "internal" state has an input term that directly affects an output state, then this input can simply be replaced by the corresponding output state and hence leaving this internal state with no direct input term. In summary,

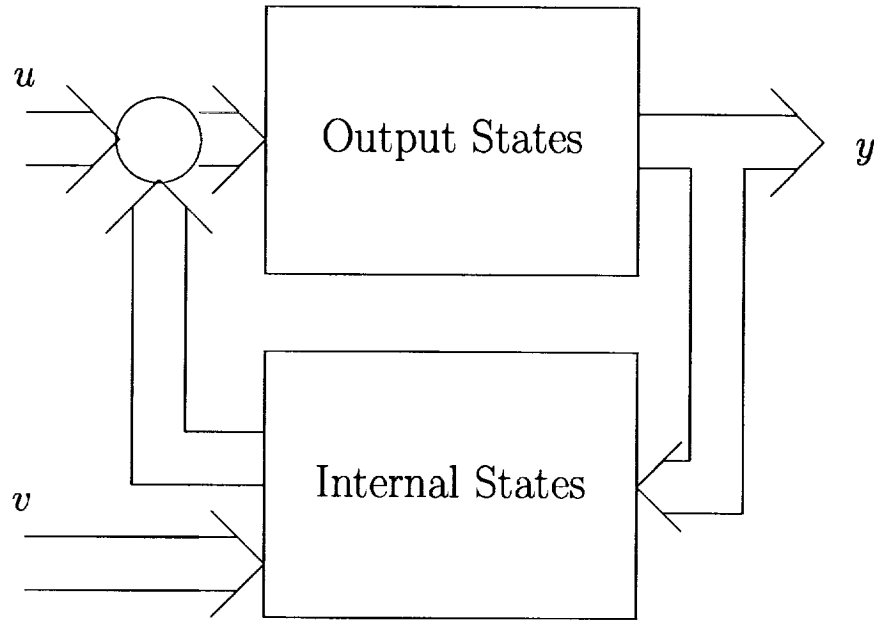


Figure 5.1: S.C.B. Diagram

Internal States		Output States	
$x_a$ : No direct input	$x_c$ : Direct input $v$ ( $u$ and $v$ disjoint)	$y_f$ : Direct input $u$	$y_b$ : No direct input

Invariant zeros are the eigenvalues of the partition of the SCB transformed system matrix corresponding to the states  $x_a$ . Left and right invertibility are easily visualized in the SCB transformed space and are not to be confused with controllability and observability. Left invertibility can be defined as follows: for a given set of outputs, there exists a unique set of inputs that generates these outputs. A system is left invertible if the states  $x_c$  are non-existent. Similarly, it is right invertible if the states  $y_b$  are non-existent.

The condition of left invertibility is not the same as that of observability. All the output states  $y_f$  and  $y_b$  are observable, so a lack of observability *may* only exist in the internal states  $x_a$  and  $x_c$ . By the same token, some of the output states  $y_b$  and/or some of the internal states  $x_a$  *may* be uncontrollable.

The SCB transformation allows designers to examine the general system *input-output properties* for a given set of inputs and outputs. It should be noted that the inputs can be either controls and/or disturbance inputs, and the outputs can be either sensor and/or criterion outputs.

In the context of closed-loop transfer recovery, conditions for either exact or asymptotic recovery depend on the system input-output properties between the disturbances  $w(t)$  and

the sensor outputs  $y_s(t)$  for the state-feedback stabilized system

$$\Sigma_{y_s w} : \begin{cases} \dot{x}(t) &= (F - GK)x(t) + Gu(t) + \Gamma w(t) \\ y_s(t) &= H_s x(t) + D_{su}u(t) + D_{sw}w(t) \end{cases} \quad (5.2)$$

This analysis applies only to recovery designs with a full or reduced-order observer-based controller. Issues related to controllability are implicitly resolved with the existence of a suitable state-feedback law. In the next section, we recall some basic conditions for recoverability obtained in [11] for observer-based controllers.

## 5.4 Conditions for Exact and Asymptotic Closed-Loop Transfer Recovery

As previously mentioned, we need to examine the SCB properties of the system  $\Sigma_{y_s w}$  between the disturbance input  $w(t)$  and the sensor output  $y_s(t)$ . It has been shown [11] that recovery is possible if this system  $\Sigma_{y_s w}$  is left invertible and has stable invariant zeros. For the case of full-order observer-based controllers, *exact* recovery is possible if furthermore the system  $\Sigma_{y_s w}$  under the SCB transformation has no  $y_f$  states; otherwise, one can only achieve asymptotic recovery. In another word, full-state feedback design is not recoverable if the system  $\Sigma_{y_s w}$  has  $x_c$  states (i.e., not left-invertible) or nonminimum phase (i.e., invariant zeros on the right-half plane).

In the case of reduced-order observer-based controllers (i.e., Luenberger observer), we have the same recovery conditions as in the case of full-order observer-based controllers. However, exact recovery is still possible even when there are output states  $y_f$ . The exact recovery condition requires simply that each output state in  $y_f$  is separated from the inputs  $u(t)$  by at most one integrator. If more than one integrator exists between them, then the recovery can only be achieved asymptotically.

A more complete discussion on recoverability of observer-based controllers can be found in [11]. For non-recoverable systems, the nonzero values of the auxiliary transfer function output  $M(s)$  are transformed to the actual error

$$E(s) = T_{zu}(s)M(s)$$

which could be quite large for a nonzero  $M(s)$ . For the non-recoverable case, the CLTR design problem is more appropriately addressed through minimization of the actual recovery error  $E(s)$ , which makes the use of numerical optimization necessary to determine the design solution. This design approach is presented in Chapter 6.

## 5.5 CLTR Design with a Full-Order Observer-Based Controller

Consider the recovery error achieved under a full-order observer-based controller [11],

$$E_f(s) = T_{zw}^{<f>}(s) - T_{zw}(s) = T_{zu}(s)M_f(s)$$

with  $T_{zu}(s)$  is the closed-loop transfer function between the control  $u(s)$  and the criterion output  $z(s)$  under the state-feedback design. Similarly,  $T_{zw}(s)$  is the closed-loop transfer

between the disturbances  $w(s)$  and the criterion output  $z(s)$ . The transfer function  $T_{zw}^{<f>}(s)$  corresponds to the closed-loop system with a full-order observer-based controller.

Consider the following plant and full-order observer-based controller,

Plant	
$\dot{x}(t)$	$= Fx(t) + \Gamma w(t) + Gu(t)$
$z(t)$	$= H_c x(t) + D_{cw} w(t) + D_{cu} u(t)$
$y_s(t)$	$= H_s x(t) + D_{sw} w(t) + D_{su} u(t)$
Full-Order Observer-Based Controller	
$\hat{\dot{x}}(t)$	$= (F - K^{obs} H_s) \hat{x}(t) + Gu(t) + K^{obs} y_s(t)$
$u(t)$	$= -K \hat{x}(t)$

where  $K$  is the state-feedback gain matrix and  $K^{obs}$  is the observer gain matrix to be determined under the CLTR procedure. The auxiliary error  $M_f(s)$  can be described directly in terms of the following state model

$$\begin{aligned} M_f(s) &= K(sI - F + K^{obs} H_s)^{-1} (\Gamma - K^{obs} D_{sw}) \\ M_f^T(s) &= (\Gamma^T - D_{sw}^T K^{obs^T}) (sI - F^T + H_s^T K^{obs^T})^{-1} K^T \end{aligned}$$

The system transfer function  $M_f^T(s)$  can be described by the following auxiliary system

$$\begin{cases} \dot{\bar{x}}(t) &= F^T \bar{x}(t) + H_s^T \bar{u}(t) + K^T \bar{w}(t) \\ \bar{z}(t) &= \Gamma^T \bar{x}(t) + D_{sw}^T \bar{u}(t) \end{cases}$$

under the state-feedback control  $\bar{u}(t) = -K^{obs^T} \bar{x}(t)$ . Synthesis of the observer gain  $K^{obs^T}$  can be performed through the minimization of the  $H^2$ -norm of the criterion output  $\bar{z}(t)$  as in an LQR state-feedback problem. Namely,

$$\min_{K^{obs}} \int_0^\infty \bar{z}^T(t) \bar{z}(t) dt = \min_{K^{obs}} \int_0^\infty \begin{bmatrix} \bar{x}^T(t) & \bar{u}^T(t) \end{bmatrix} \begin{bmatrix} \Gamma \Gamma^T & \Gamma D_{sw}^T \\ D_{sw} \Gamma^T & D_{sw} D_{sw}^T \end{bmatrix} \begin{bmatrix} \bar{x}(t) \\ \bar{u}(t) \end{bmatrix} dt$$

This problem is a *singular*  $H^2$ -optimal control due to the fact that the weighting matrix  $D_{sw} D_{sw}^T$  is usually singular. One can solve this problem by introducing a small perturbation into the singular portion of this matrix. To do this, we first perform a singular value decomposition on  $D_{sw}$  as follows,

$$D_{sw} = U_D \Sigma_D V_D^T \Rightarrow D_{sw} D_{sw}^T = U_D \Sigma_D \Sigma_D^T U_D^T$$

where

$$\Sigma_D \Sigma_D^T = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \cdots & & \\ 0 & \sigma_2^2 & 0 & \cdots & & \\ \vdots & & & \ddots & & \\ 0 & \cdots & & & \sigma_n^2 & \cdots & 0 \\ 0 & \cdots & & & & 0 & \\ 0 & \vdots & & & & & \ddots \end{bmatrix}$$

Singularity of a matrix is usually defined relative to its maximum singular value as in the definition of the condition number. Let's choose a small perturbation equal to  $\epsilon \sigma_1^2$  with  $\epsilon$  in

the vicinity of the machine precision. If this number is greater than  $\sigma_m^2$  for some  $m$ , then one would perturb the matrix  $D_{sw}D_{sw}^T$  by

$$R' = [D_{sw}D_{sw}^T]_{\text{perturbed}} = D_{sw}D_{sw}^T + \epsilon\sigma_1^2 U_D \begin{bmatrix} [0]_{(m-1) \times (m-1)} & 0 \\ 0 & [I] \end{bmatrix} U_D^T$$

This problem can be solved as a *regular* LQR problem and solution obtained from an algebraic Riccati equation

$$0 = S \left( F^T - H_s^T R'^{-1} D_{sw} \Gamma^T \right) + \left( F - \Gamma D_{sw}^T R'^{-1} H_s \right) S + \Gamma \left( I - D_{sw} R'^{-1} D_{sw} \Gamma^T \right) \Gamma^T - S H_s^T R'^{-1} H_s S$$

with  $K^{obs} = R'^{-1} (H_s S + D_{sw} \Gamma^T)$ . Let's define  $E(s, \epsilon) = T_{zw}^{<f>}(s, \epsilon) - T_{zw}(s)$  associated with the family of closed-loop transfer functions  $T_{zw}^{<f>}(s, \epsilon)$ . One can examine the asymptotic properties as  $\epsilon \rightarrow 0$ . If the conditions for exact recovery are satisfied, then for  $\epsilon \rightarrow 0$ , the gain matrix  $K^{obsT}$  will converge to a finite gain matrix as  $E(s, \epsilon) \rightarrow 0$ . Otherwise, some of the gains in  $K^{obs}$  will approach infinity. Closed-loop responses of the state-feedback design associated with infinite zeros of the system  $\Sigma_{y_s w}$  will not be recovered asymptotically while those corresponding to the finite stable invariant zeros will be exactly recovered. In the case of asymptotic recovery, although  $E(s, \epsilon) \rightarrow 0$  as  $\epsilon \rightarrow 0$ , one must pick an  $\epsilon$  representing a desired level of recovery along with a reasonable finite gain for  $K^{obs}$ .

## 5.6 CLTR with a Luenberger Observer

In the Luenberger observer design, some of the measurement outputs are assumed to be noise free. For a given state-feedback matrix  $K$ , a controller based on the Luenberger observer is of the form,

$$\begin{aligned} \dot{v}(t) &= Lv(t) + M_1 y_s(t) + M_2 u(t) \\ \hat{x}(t) &= Pv(t) + Jy_s(t) \\ u(t) &= -K\hat{x}(t) \end{aligned}$$

with

$$QF - LQ = M_1 H_s, \quad M_2 = QG,$$

and

$$JH_s + PQ = I_n$$

Note that  $n$  is the order of the plant model.

Without loss of generality, we can always transform the plant model such that the noise-free outputs of  $y_s(t)$  become a subset of the system states (see [11], Section 4.2).

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} u(t) + \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} w(t) \\ z(t) &= H_c x(t) + D_{cw} w(t) + D_{cu} u(t) \\ y_s(t) = \begin{bmatrix} y_0(t) \\ y_1(t) \end{bmatrix} &= \begin{bmatrix} 0 & H_{s,02} \\ I_{p-m_0} & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} D_{sw,0} \\ 0 \end{bmatrix} w(t) \end{aligned}$$

Clearly, an observer is needed to estimate the states  $x_2(t)$ . Defining an observer gain matrix  $K_r^{obs} = [K_{r0}^{obs}, K_{r1}^{obs}]$  whose partitions correspond to the measurement outputs  $y_0(t)$  and  $y_1(t)$  respectively, we obtain a controller with a reduced-order Luenberger observer of the form [12],

$$\begin{aligned}\dot{v}(t) &= A_{or}v(t) + (G_2 - K_{r1}^{obs}G_1)u(t) + K_{r0}^{obs}y_0(t) \\ &\quad + (F_{21} - K_{r1}^{obs}F_{11} + A_{or}K_{r1}^{obs})y_1(t) \\ \hat{x}(t) &= \begin{bmatrix} 0 \\ I_{n-p+m_0} \end{bmatrix} v(t) + \begin{bmatrix} I_{p-m_0} \\ K_{r1}^{obs} \end{bmatrix} y_1(t) \\ u(t) &= -K\hat{x}(t)\end{aligned}$$

where

$$A_{or} = F_{22} - K_{r0}^{obs}H_{s,02} - K_{r1}^{obs}F_{12}$$

We also partition the state feedback gain matrix  $K = [K_1, K_2]$  in accordance with the state partitions  $x_1(t)$  and  $x_2(t)$ . The recovery error can then be written as in [11],

$$E_r(s) = T_{zw}^{<r>}(s) - T_{zw}(s) = T_{zu}(s)M_r(s)$$

where  $T_{zw}^{<r>}(s)$  is a closed-loop transfer function incorporating a reduced-order observer-based controller, and

$$M_r(s) = K_2 (sI - A_r + K_r^{obs}C_r)^{-1} (B_r - K_r^{obs}D_r)$$

or

$$M_r^T(s) = (B_r^T - D_r^T K_r^{obsT}) (sI - A_r^T + C_r^T K_r^{obsT})^{-1} K_2^T$$

where

$$\begin{aligned}A_r &= F_{22} \quad , \quad B_r = \Gamma_2, \\ C_r &= \begin{bmatrix} H_{s,02} \\ F_{12} \end{bmatrix}, \quad D_r = \begin{bmatrix} D_{sw,0} \\ \Gamma_1 \end{bmatrix}\end{aligned}$$

As in the case of full-order observer-based controller, an auxiliary system can be constructed such that its closed-loop transfer function under a “state-feedback” law is equal to  $M_r^T(s)$ . It is given by

$$\begin{cases} \dot{\bar{x}}_r(t) &= A_r^T \bar{x}_r(t) + C_r^T \bar{u}(t) + K_2^T \bar{w}(t) \\ \bar{z}(t) &= B_r^T \bar{x}_r(t) + D_r^T \bar{u}(t) \end{cases}$$

and  $\bar{u}(t) = -K_r^{obs} \bar{x}_r(t)$ .

The reduced-order observer design gain matrix  $K_r^{obs}$  can be obtained from the minimization of the  $H^2$ -norm of the criterion output  $\bar{z}(t)$ . If the associated LQR problem is singular, one can perturb the weighting matrix  $D_r D_r^T$  in a design approach similar to the that presented in Section 5.5. As in the full-order observer case, closed-loop responses corresponding to finite stable invariant zeros will be exactly recovered. Furthermore, responses associated with first-order infinite zeros will also be exactly recovered. Responses corresponding to infinite zeros of second and higher order will be asymptotically recovered.

## 5.7 Conclusions

In this chapter we introduced the concept of closed-loop transfer recovery. Analysis of recoverability using observer-based controllers can be done using system properties derived from the SCB transformation. A procedure based on Riccati solutions is given for the design of full and reduced-order observer gain matrices.

## Chapter 6

# CLTR Using Numerical Optimization

### 6.1 Introduction

Although numerical optimization provides a powerful design tool for the synthesis of multi-variable control systems, its usage is still limited. Difficulties often encountered in optimal control synthesis are related to how one formulates the design problem to account for all the basic requirements in stability, performance and robustness. Even in  $H^2/H^\infty$ -optimal control, extensive design trade-offs are often needed before a satisfactory optimum design can be found. Some of the trade-offs reside in the selection of a quadratic performance index. Here, one needs to identify, through successive trials, the appropriate design criterion variables, the control variables, and their relative effects on the overall design. The process tends to be computationally intensive and time consuming if one considers direct synthesis of low-order controllers using numerical optimization. However, if the control problem can first be solved under the setting of state feedback whose solutions can easily be obtained from the algebraic Riccati equation, one can then proceed to the design of output-feedback controllers using the concept of closed-loop transfer recovery. The merits of this approach are that few design iterations are needed in obtaining an output-feedback controller, especially when a low-order controller is derived from the order reduction of an observer-based controller that achieves exact recovery.

In this chapter, we address the problem of closed-loop transfer recovery using an arbitrary controller structure. As indicated in the previous chapter, analysis and design of closed-loop transfer recovery are well developed in the setting of observer-based controllers [11]-[12]. These results form the baseline designs to CLTR using numerical optimization. Performance and robustness achieved under observer-based controllers serve as guidelines in defining a satisfactory output-feedback design of lower order. A tentative low-order controller design can be obtained from order reduction of an observer-based controller using a balanced truncation method. Hankel singular values of the observer-based controller model can be used to determine the possible order of the dynamic compensator in numerical optimization. The advantage of numerical optimization is that it allows designers to seek controllers of a specific (practical) order and structure in the recovery of the closed-loop characteristics achieved under state feedback. The numerical scheme can further be used to re-optimize the recovery error over a given control/command bandwidth using frequency-shaped filters.

## 6.2 Problem Formulation

In this section, we define the problem associated with closed-loop recovery using an arbitrary dynamic compensator. Under this setting, there is no longer the simple relation for the recovery error as defined in equation 5.1.

For a dynamic compensator of the form

$$\Sigma_c : \begin{cases} \dot{x}_c(t) &= A_c x_c(t) + B_c y_s(t) \\ u(t) &= C_c x_c(t) + D_c y_s(t) \end{cases}$$

analytical results for closed-loop transfer recovery analysis and design are not possible in general. Closed-loop transfer recovery problems presented in [11] and [12] apply only to observer-based controllers. Given a recoverable system for an observer-based design, a controller with the general structure shown above is also likely to recover the desired state-feedback properties.

With  $H^2$  optimal control,  $H^\infty$  optimization, pole placement, or other design techniques, one can determine a set of state-feedback gains  $K$  that meet desired closed-loop characteristics through the control  $u(t) = -Kx(t)$ . The resulting closed-loop system

$$\Sigma_{zw} : \begin{cases} \dot{x}_f(t) &= (F - GK)x_f(t) + \Gamma w(t) \\ z(t) &= (H_c - D_{cu}K)x_f(t) + D_{cw}w(t) \end{cases}$$

defines the ideal transfer function  $T_{zw}(s)$  to be recovered with an output-feedback design.

In the design approach based on direct optimization for CLTR, the quantity being minimized is the actual error

$$E(s) = T_{zw}(s) - T_{zw}^o(s)$$

rather than the auxiliary transfer function  $M_f(s)$  (equation 5.1). Typically the disturbance input  $w(t)$  is modelled as white noise. Frequency shaping can be introduced into  $w(t)$  using additional filter dynamics

$$\Sigma_w : \begin{cases} \dot{x}_w(t) &= F_w x_w(t) + \Gamma_w \eta(t) \\ w(t) &= H_w x_w(t) + D_w \eta(t) \end{cases}$$

where  $\eta(t)$  are white noises. The overall synthesis model, including the noise shaping filters, the state-feedback target system dynamics, the open-loop plant, and the controller

dynamics is given by

$$\Sigma_e : \left\{ \begin{array}{l} \begin{bmatrix} \dot{x}_w(t) \\ \dot{x}_f(t) \\ \dot{x}(t) \\ \dot{x}_c(t) \end{bmatrix} = \begin{bmatrix} F_w & 0 & 0 & 0 \\ \Gamma H_w & F - GK & 0 & 0 \\ \Gamma H_w & 0 & F & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_w(t) \\ x_f(t) \\ x(t) \\ x_c(t) \end{bmatrix} + \begin{bmatrix} \Gamma_w \\ \Gamma D_w \\ \Gamma D_w \\ 0 \end{bmatrix} \eta(t) \\ \quad + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ G & 0 \\ 0 & I_r \end{bmatrix} \begin{bmatrix} u \\ u_c \end{bmatrix} \\ \\ e(t) = \begin{bmatrix} 0 & (H_c - D_{cu}K) & -H_c & 0 \end{bmatrix} \begin{bmatrix} x_w(t) \\ x_f(t) \\ x(t) \\ x_c(t) \end{bmatrix} - \begin{bmatrix} D_{cu} & 0 \end{bmatrix} \begin{bmatrix} u(t) \\ u_c(t) \end{bmatrix} \\ \\ y_e(t) = \begin{bmatrix} D_{sw}H_w & 0 & H_s & 0 \\ 0 & 0 & 0 & I_r \end{bmatrix} \begin{bmatrix} x_w(t) \\ x_f(t) \\ x(t) \\ x_c(t) \end{bmatrix} + \begin{bmatrix} D_{sw}D_w \\ 0 \end{bmatrix} \eta(t) \\ \quad + \begin{bmatrix} D_{su} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u(t) \\ u_c(t) \end{bmatrix} \end{array} \right.$$

where  $e(t)$  is the recovery error, and  $y_e(t)$  is the measurement output vector augmented with the states of a dynamic output-feedback compensator of order  $r$ . Feedback control is conveniently implemented in a static output feedback form

$$\begin{bmatrix} u(t) \\ u_c(t) \end{bmatrix} = C_o y_e(t) = \begin{bmatrix} D_c & C_c \\ B_c & A_c \end{bmatrix} y_e(t).$$

The CLTR problem for a dynamic output-feedback compensator reduces to finding a gain matrix  $C_o$  that minimizes a certain norm of the recovery error, or

$$\min_{C_o} J_{cltr}(C_o, t_f).$$

For the  $H^2$  norm of the recovery error, the disturbance inputs are treated as white noises with  $E[\eta(t)\eta(\tau)^T] = W_o\delta(t - \tau)$ . The corresponding  $H^2$  cost is given by

$$J_{cltr} = \lim_{t_f \rightarrow \infty} E[e^T(t_f)e(t_f)].$$

By Parseval's theorem, it has the equivalent form

$$J_{cltr} = \frac{1}{\pi} \int_0^\infty \left\| [T_{zw}(j\omega) - T_{zw}^o(j\omega)] W_o^{\frac{1}{2}} \right\|^2 d\omega$$

in the frequency domain. It can be extended to include a frequency weighting  $W(j\omega)$  as follows

$$J_{cltr} = \frac{1}{\pi} \int_0^\infty \left\| [T_{zw}(j\omega) - T_{zw}^o(j\omega)] W(j\omega) \right\|^2 d\omega.$$

Due to the difficulty in finding an analytic solution for a general controller structure, a minimum  $J_{cltr}$  is often obtained by nonlinear optimization techniques.

Difficulties in numerical optimization for the CLTR approach are associated with the following two factors. The first is that poles of the closed-loop system in the output-feedback design are optimized toward those of the “reference” state-feedback system. While the eigenvalues are coming closer together, the modes themselves remain distinct because the dynamics of the state-feedback and the output-feedback systems contained in the synthesis model remain distinct. The typically large size of the overall system matrix brings in the second factor. Roundoff in computations involving large system matrices could cause non-degenerate modes with the same eigenvalue to appear degenerate. With degenerate modes, or correspondingly, defective eigenvalues in the system matrix, algorithms for computing the cost function and its gradient based on an eigenvalue-eigenvector decomposition will break down. Since a degenerate system matrix is likely in the CLTR procedure for most practical problems, one must employ techniques for computing the cost and its gradient that are immune to this condition.

### 6.3 Feedforward Controller in the CLTR Design Procedure

We present a design approach for feedforward gains in numerical CLTR. Determination of the feedforward gains in the optimization process would certainly provide the best responses. However, because the feedforward gains do not affect stability, one need not include them in the controller optimization process, thus reducing the number of parameters involved and saving computation time.

A particular feedforward structure is chosen here for convenience. Given a feedback controller design, the basic idea is to construct a feedforward gain to achieve the desired steady-state relations between command inputs and the commanded outputs. Although we limit the scope to constant gain matrices, the resulting solutions are often non-unique. First and foremost, we review the computation of feedforward gains for the state feedback case. In the subsequent sections, we consider observer-based controllers followed by the case for a general controller structure.

#### 6.3.1 A Feedforward Controller under State-Feedback

This is the simplest case of feedforward controller design. We write the system dynamics and commanded output in a single set of equations

$$\begin{bmatrix} \dot{x}(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} (F - GK) & G \\ (C_{cmd} - D_{cmd}K) & D_{cmd} \end{bmatrix} \begin{bmatrix} x(t) \\ u_{cmd}(t) \end{bmatrix} \quad (6.1)$$

Here  $u_{cmd}(t)$  is the direct control input derived only from the command feedforward gains acting upon  $y_{ref}(t)$ . A feedforward controller is then determined such that the system outputs  $y_{cmd}(t)$  match the command reference inputs  $y_{ref}(t)$  in steady-state, i.e., when  $\dot{x}(t) = 0$ . Solving equation (6.1) for steady-state values of the states and the control inputs, we have

$$\begin{bmatrix} x_{ss} \\ u_{cmd} \end{bmatrix} = \begin{bmatrix} (F - GK) & G \\ (C_{cmd} - D_{cmd}K) & D_{cmd} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ y_{ref} \end{bmatrix} \quad (6.2)$$

or

$$\begin{bmatrix} x_{ss} \\ u_{cmd} \end{bmatrix} = \begin{bmatrix} L \\ N \end{bmatrix} y_{ref}$$

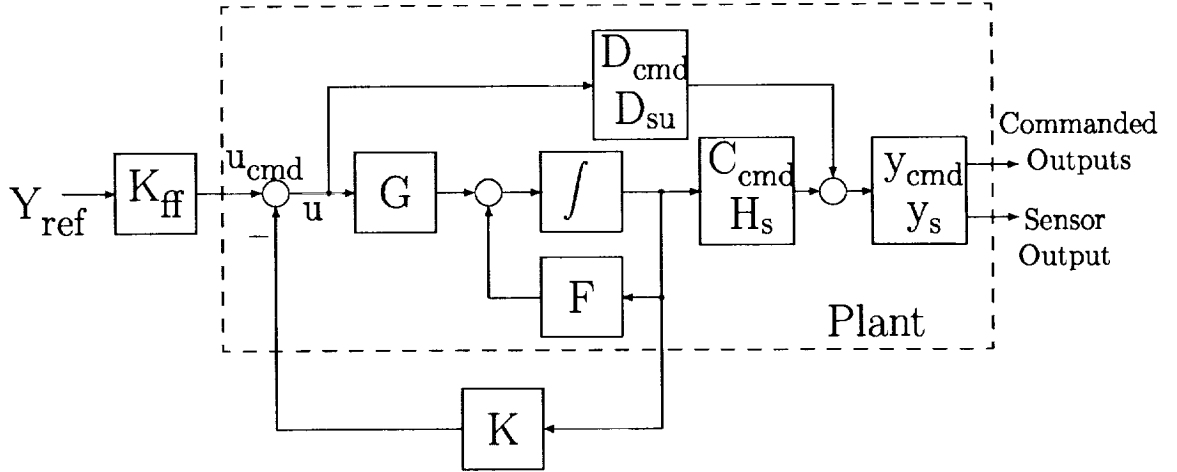


Figure 6.1: State Feedback System with a Feedforward Gain

where  $x_{ss}$  is the steady-state value of  $x$ , and

$$\begin{bmatrix} L \\ N \end{bmatrix} = \begin{bmatrix} (F - GK) & G \\ (C_{cmd} - D_{cmd}K) & D_{cmd} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}$$

From this, we can deduce that  $u_{cmd}(t) = Ny_{ref}(t)$  and therefore  $K_{ff} = N$ . Very often the matrix

$$\begin{bmatrix} (F - GK) & G \\ (C_{cmd} - D_{cmd}K) & D_{cmd} \end{bmatrix}$$

is not full rank, therefore we must take the pseudoinverse rather than the actual inverse. This signals the existence of more than one solution to the feedforward gains (we have more degrees of freedom than necessary). This feature comes in use when we pick convenient solutions to some of the cases that follow.

Considerable simplification can happen when one considers command inputs to integral states. There are two similar, but distinct cases involving integral states recognized here. In the first case, the integral states are part of the plant model

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}^I(t) \end{bmatrix} = \begin{bmatrix} F & 0 \\ F^I & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x^I(t) \end{bmatrix} + \begin{bmatrix} G \\ 0 \end{bmatrix} u(t).$$

Assume there is no direct control input into either the integral states or the commanded output  $y_{cmd}$  ( $G^I = 0$  and  $D_{cmd} = 0$ ). Partitioning the state feedback gain matrix into  $[K \ K^I]$ , the overall system would look like

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}^I(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} F - GK & -GK^I & G \\ F^I & 0 & 0 \\ 0 & C_{cmd} & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x^I(t) \\ u_{cmd}(t) \end{bmatrix}. \quad (6.3)$$

Again, the steady-state solution would correspond to  $\dot{x} = 0$ ,  $\dot{x}^I = 0$ , and  $y_{cmd} = y_{ref}$ . Let's consider an alternative solution  $u_{cmd} = K^I C_{cmd}^{-1} y_{ref}$  or  $K_{ff} = K^I C_{cmd}^{-1}$ . (It is assumed, without loss of generality, that  $C_{cmd}$  is full rank). With this feedforward gain, we have

$x_{ss} = 0$ , and  $x_{ss}^I = C_{cmd}^{-1}y_{ref}$ . Other nontrivial solutions (nonzero  $x_{ss}$ ) of equation (6.3) are possible.

The second case involves plant dynamics that are augmented with the integral states  $\dot{x}^I(t) = y_{cmd}(t) - y_{ref}(t)$ . The combined equations for the augmented system dynamics and the commanded output are given by

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}^I(t) + y_{ref}(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} F - GK & -GK^I & G \\ C_{cmd} & 0 & D_{cmd} \\ C_{cmd} & 0 & D_{cmd} \end{bmatrix} \begin{bmatrix} x(t) \\ x^I(t) \\ u_{cmd}(t) \end{bmatrix} \quad (6.4)$$

In steady-state, we have

$$\begin{bmatrix} F - GK & -GK^I & G \\ C_{cmd} & 0 & D_{cmd} \\ C_{cmd} & 0 & D_{cmd} \end{bmatrix} \begin{bmatrix} x_{ss} \\ x_{ss}^I \\ u_{cmd} \end{bmatrix} = \begin{bmatrix} 0 \\ y_{ref} \\ y_{ref} \end{bmatrix}$$

or

$$\begin{bmatrix} F - GK & -GK^I & G \\ C_{cmd} & 0 & D_{cmd} \end{bmatrix} \begin{bmatrix} x_{ss} \\ x_{ss}^I \\ u_{cmd} \end{bmatrix} = \begin{bmatrix} 0 \\ y_{ref} \end{bmatrix}$$

While this can be solved for  $K_{ff}$  as before for  $u_{cmd} = K_{ff}y_{ref}$ , it must be noted that the command path through the integral states will govern the long-term command response. The structure of the above equations guarantees the existence of more than one nontrivial solution ( $x_{ss} \neq 0$ ) for  $K_{ff}$  when such a solution exists, allowing a choice of transient responses to command inputs. Often  $K_{ff}$  is set to zero.

### 6.3.2 A Feedforward Controller with Observer-Based Controllers

Design of the feedforward gain matrices  $K_{ff}$  and  $G_c$  (as shown in Figure 6.2) will differ slightly between a feedback controller design based on an observer-based controller structure and one with an arbitrary dynamic compensator structure.

A direct feedforward input into the controller dynamics provides an “anticipation” factor to the command inputs. An observer-based controller using the CLTR procedure possesses an inherent feedforward controller structure. This is due primarily to the fact that the observer has a direct input term from the control  $u(t)$ . Let’s examine in more detail the influence of a command input in an observer-based controller design. The observer-based controller is given by

$$\begin{aligned} \dot{x}_c(t) &= A_c x_c(t) + B_c y_s(t) + G_2 u(t) \\ \hat{x}(t) &= P x_c(t) + J y_s(t) \end{aligned}$$

where the control output is

$$\begin{aligned} u(t) &= K \hat{x}(t) + u_{cmd}(t) \\ &= K P x_c(t) + K J y_s(t) + u_{cmd}(t) \\ &= C_c x_c(t) + D_c y_s(t) + u_{cmd}(t) \\ &= C_c x_c(t) + D_c [H_s x(t) + D_{su} u(t)] + u_{cmd}(t) \\ &= C_c x_c(t) + D_c H_s x(t) + (I + D_c D_{su}) u_{cmd}(t). \end{aligned}$$

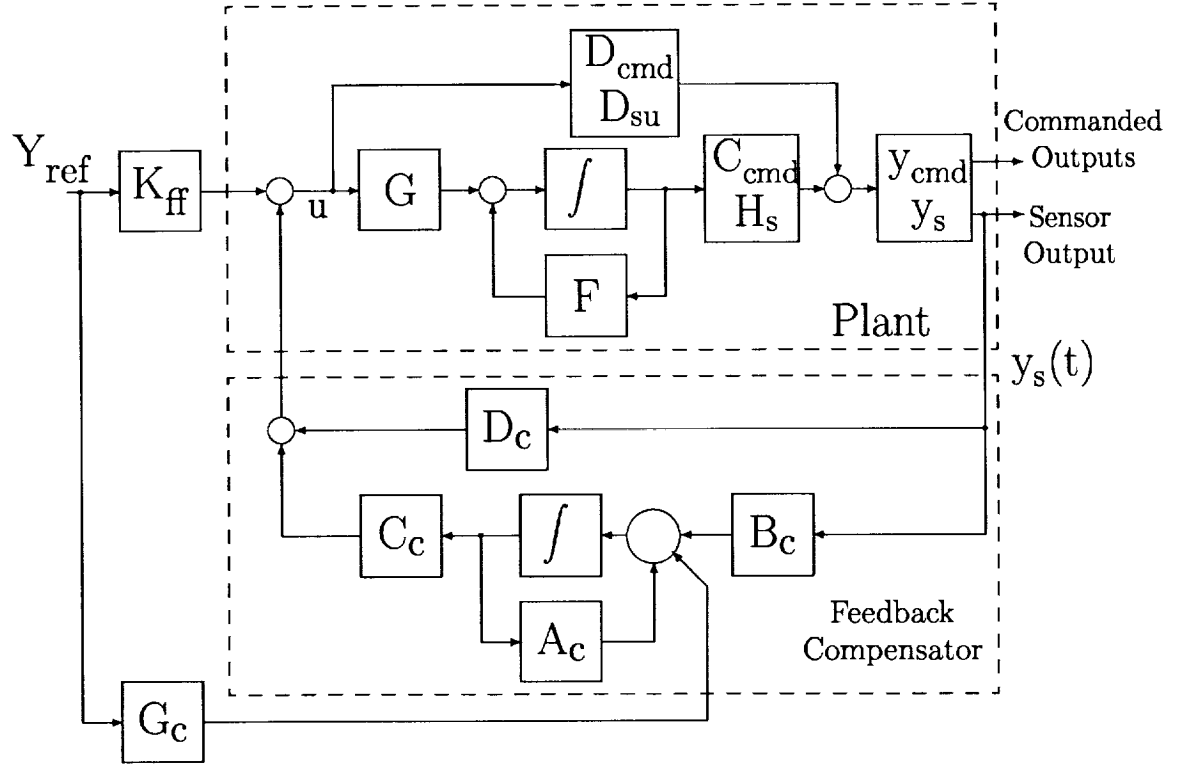


Figure 6.2: Closed-Loop System with a Feedback/Feedforward Controller

where we define  $C_c = KP$  and  $D_c = KJ$ . For simplicity, we assume  $D_c D_{su} = 0$ . The combined closed-loop dynamics and command output equations are given by

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_c(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} (F + GD_c H_s) & GC_c & G \\ \begin{bmatrix} B_c(I + D_{su} D_c) \\ +G_2 D_c \end{bmatrix} H_s & A_c + B_c D_{su} C_c + G_2 C_c & B_c D_{su} + G_2 D_{cmd} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix} \begin{bmatrix} x \\ x_c \\ u_{cmd} \end{bmatrix} \quad (6.5)$$

In steady state, we have

$$\begin{bmatrix} x_{ss} \\ x_{css} \\ u_{cmd} \end{bmatrix} = \begin{bmatrix} (F + GD_c H_s) & GC_c & G \\ \begin{bmatrix} B_c(I + D_{su} D_c) \\ +G_2 D_c \end{bmatrix} H_s & A_c + B_c D_{su} C_c + G_2 C_c & B_c D_{su} + G_2 D_{cmd} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ y_{ref} \end{bmatrix}$$

or

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} (F + GD_cH_s) & GC_c & G \\ [B_c(I + D_{su}D_c) + G_2D_c]H_s & A_c + B_cD_{su}C_c + G_2C_c & B_cD_{su} + G_2D_{cmd} \\ (C_{cmd} + D_{cmd}D_cH_s) & D_{cmd}C_c & D_{cmd} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}$$

We have  $u_{cmd}(t) = Ny_{ref}(t)$  or  $K_{ff} = N$ . Within the context of Figure 6.2, it is straightforward to see that  $G_c = G_2K_{ff}$ .

Like the state-feedback case, some simplification is possible if one considers problems with integral control

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}^I(t) \end{bmatrix} = \begin{bmatrix} F & 0 \\ C_{cmd} & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x^I(t) \end{bmatrix} + \begin{bmatrix} G \\ 0 \end{bmatrix} u(t).$$

Note that the control is given by  $u(t) = C_c x_c(t) + D_c y_s(t) + D_c^I x^I(t)$ . This would result in a closed-loop system of the form

$$\begin{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \dot{x}_c(t) \end{bmatrix} \\ \dot{x}^I(t) + y_{ref}(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} F_{cl} + \begin{bmatrix} G \\ G_2 \end{bmatrix} [D_c H_s \ C_c] & \begin{bmatrix} G \\ G_2 \end{bmatrix} D_c^I & \begin{bmatrix} G \\ G_2 \end{bmatrix} \\ \begin{bmatrix} C_{cmd} & 0 \\ C_{cmd} & 0 \end{bmatrix} & 0 & \begin{bmatrix} D_{cmd} \\ D_{cmd} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix} \\ x^I(t) \\ u_{cmd}(t) \end{bmatrix}$$

where

$$F_{cl} = \begin{bmatrix} F & 0 \\ B_c(I + D_{su}D_c)H_s & A_c + B_cD_{su}C_c \end{bmatrix}$$

As in equation (6.4) for the state feedback case, the above form allows more than one solution to the command feedforward  $u_{cmd} = K_{ff}y_{ref}$ , with  $K_{ff} = 0$  often being used. As before, only the short-term response is governed by this feedforward.

### 6.3.3 A Feedforward Controller with Non-Observer-Based Controllers

We can now proceed to the design of the feedforward controller for an arbitrary feedback compensator of the form,

$$\begin{aligned} \dot{x}_c(t) &= A_c x_c(t) + B_c y_s(t) \\ u(t) &= C_c x_c(t) + D_c y_s(t) \end{aligned}$$

The overall closed-loop structure is exactly as shown in Figure 6.2. Because there is no pre-set distribution from the control  $u_{cmd}$  to the controller, determination of the feedforward gains  $K_{ff}$  and  $G_c$  will have more degrees of freedom due to the nominal independence of the two matrices.

We will cover three common approaches for designing the feedforward gain matrices  $K_{ff}$  and  $G_c$ . In the most general case, one simultaneously creates both matrices. We first write state and output equations for the overall closed-loop system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_c(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} (F + GD_cH_s) & GC_c & G & 0 \\ B_c(I + D_{su}D_c)H_s & (A_c + B_cD_{su}C_c) & B_cD_{su} & I \\ (C_{cmd} + D_{cmd}D_cH_s) & D_{cmd}C_c & D_{cmd} & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \\ u_{cmd}(t) \\ v_{cmd}(t) \end{bmatrix}$$

where  $v_{cmd}(t) = G_c y_{ref}(t)$  and we again assume  $D_c D_{su} = 0$ . The above system is not square, with more degrees of freedom than equations. Thus, one can solve the above using a singular value decomposition on the system matrix

$$U_F \Sigma_F V_F^T = \begin{bmatrix} (F + G D_c H_s) & G C_c & G & 0 \\ B_c(I + D_{su} D_c) H_s & (A_c + B_c D_{su} C_c) & B_c D_{su} & I \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} & 0 \end{bmatrix}$$

by constructing a pseudoinverse. The form of the pseudoinverse is  $V_F \Sigma_F^\dagger U_F^T$ , where  $\Sigma_F^\dagger$  is generated by transposing the singular value matrix  $\Sigma_F$  and replacing each nonzero singular value with its reciprocal. Applying this to the steady-state response,

$$\begin{bmatrix} x_{ss} \\ x_{css} \\ u_{cmd} \\ v_{cmd} \end{bmatrix} = V_F \Sigma_F^\dagger U_F^T \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} y_{ref} = \begin{bmatrix} L \\ M \\ K_{ff} \\ G_c \end{bmatrix} y_{ref}.$$

The other two methods depend on pre-selecting either of the matrices  $K_{ff}$  or  $G_c$ . In direct optimization of a controller, one often has the commanded outputs coinciding with several of the sensed outputs, in which case it would be reasonable to make  $-G_c$  equal to selected columns from  $B_c$ . The overall system becomes

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_c(t) - G_c y_{ref}(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} (F + G D_c H_s) & G C_c & G \\ B_c(I + D_{su} D_c) H_s & (A_c + B_c D_{su} C_c) & B_c D_{su} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \\ u_{cmd}(t) \end{bmatrix}$$

or

$$\begin{bmatrix} L \\ M \\ K_{ff} \end{bmatrix} = \begin{bmatrix} (F + G D_c H_s) & G C_c & G \\ B_c(I + D_{su} D_c) H_s & (A_c + B_c D_{su} C_c) & B_c D_{su} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -G_c \\ I \end{bmatrix}$$

The pre-selection of  $K_{ff}$  happens in the context of CLTR. Since one is trying to recover state-feedback characteristics, the steady-state controller response should be like that from the state-feedback gains. Therefore it is appropriate to use the  $K_{ff}$  from state-feedback, with the remaining feedforward gain matrix  $G_c$  derived from the system

$$\begin{bmatrix} \dot{x}(t) - K_{ff} y_{ref}(t) \\ \dot{x}_c(t) \\ y_{cmd}(t) \end{bmatrix} = \begin{bmatrix} (F + G D_c H_s) & G C_c & G \\ B_c(I + D_{su} D_c) H_s & (A_c + B_c D_{su} C_c) & B_c D_{su} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \\ v_{cmd}(t) \end{bmatrix}$$

or, in solving this (usually) nonsquare system for the steady-state response,

$$U_F \Sigma_F V_F^T = \begin{bmatrix} (F + G D_c H_s) & G C_c & G \\ B_c(I + D_{su} D_c) H_s & (A_c + B_c D_{su} C_c) & B_c D_{su} \\ (C_{cmd} + D_{cmd} D_c H_s) & D_{cmd} C_c & D_{cmd} \end{bmatrix}$$

$$\begin{bmatrix} L \\ M \\ G_c \end{bmatrix} = V_F \Sigma_F^\dagger U_F^T \begin{bmatrix} -K_{ff} \\ 0 \\ I \end{bmatrix}$$

with  $\Sigma_F^\dagger$  as previously defined.

## 6.4 Concluding Remarks

In output feedback design, consideration of closed-loop stability and disturbance rejection is done through careful definition of the input model. Under closed-loop transfer recovery, these design issues are first addressed under the setting of state feedback design. Many tradeoffs can be explored quickly using a state-feedback design, especially when compared to the long turnaround associated with the direct optimization of an output-feedback controller.

Direct optimization of an output-feedback controller may not include determination of command feedforward gains. Several useful approaches are given to compute feedforward gains for an output-feedback controller optimized using the CLTR approach.

## Chapter 7

# Control Design for a High-Performance Rotorcraft via CLTR

### 7.1 Introduction

High-performance rotorcraft controller design is characterized by having to compensate for both the longitudinal and lateral dynamics in a single controller. Rather than having these modes largely decoupled in the natural state, one often needs to incorporate mode decoupling as a part of the controller design, especially when the system model includes high-frequency dynamics from the rotor flap and lag states. In addition to the usual design considerations of stability augmentation, a high-performance rotorcraft such as the UH-60 would also require good bandwidth and decoupling in the heave, yaw rate, pitch, and roll command responses.

The initial design step in CLTR is to formulate and synthesize a state-feedback controller that addresses all the design specifications. The synthesis consists of selecting appropriate design weights in the usual LQ performance index. For the UH-60 rotorcraft we construct an integral control, and incorporate a desired set of command input models. With integral control on all four important quantities—pitch, roll, yaw rate, and heave rate, we achieve the desired attitude and rate control of the respective output command variables as specified in ADS-33C for a rotorcraft.

A series of output-feedback designs are then developed from this state-feedback design using the procedure of Closed-Loop Transfer Recovery (CLTR). Initially, a Luenberger observer-based controller is designed by minimizing the difference between its closed loop response and the target closed-loop transfer function achieved with the state-feedback controller. The design minimization problem can be solved analytically through the solution of an algebraic Riccati equation. Note that in the recovery design process, we no longer need to iterate on the design performance index. With varying degrees of success in maintaining satisfactory closed-loop performance and robustness, other output-feedback designs are then derived from this controller using model reduction techniques. A numerical procedure is also used to design a CLTR controller following the problem description in Section 7.11. The controller is synthesized using one of these reduced-order designs as a starting point.

On the other hand, one can also synthesize a reduced-order controller directly using the technique described in Chapter 2. Conveniently, one of the reduced-order designs devel-

oped from the Luenberger observer-based controller can be used as a starting point in the direct optimization. The resulting optimized design will be compared to the CLTR-based controllers.

Design performance and robustness are evaluated based on the specifications defined in ADS-33C for a rotorcraft.

## 7.2 The UH-60 Model

The UH-60 rotorcraft model considered in this study is derived from a nonlinear simulation model linearized about two flight conditions: hover (nominal with 1-knot forward velocity) and a 15-knot forward velocity condition. In both cases the gross weight is set at 16,800 lbs, the rotor speed at 27 rad/s, and the air density at 0.002030 slug/cubic foot. The linearized open-loop model is of 31<sup>st</sup> order. A listing of the rotorcraft states is given in Table 7.1.

Table 7.1: UH-60 Rotorcraft States

$p, q, r$	Body-axis attitude rates (deg/s)
$u, v, w$	Body-axis velocities (ft/s)
$\theta_x, \theta_y, \theta_z$	Euler angles (deg) from terrestrial to rotorcraft axes (pitch, roll, yaw)
$x, y, z$	Inertial positions (ft)
$\dot{\beta}_0, \dot{\beta}_D, \dot{\beta}_{1S}, \dot{\beta}_{1C}$	Rotor flapping rates (deg/s)
$\dot{\zeta}_0, \dot{\zeta}_D, \dot{\zeta}_{1S}, \dot{\zeta}_{1C}$	Rotor lead-lag rates (deg/s)
$\beta_0, \beta_D, \beta_{1S}, \beta_{1C}$	Rotor flapping angles (deg)
$\zeta_0, \zeta_D, \zeta_{1S}, \zeta_{1C}$	Rotor lead-lag angles (deg)
$\lambda_0, \lambda_{1S}, \lambda_{1C}$	Inflow velocities (1/sec) normalized by the rotor radius (26.83ft)

The open-loop linear models in both the hover and 15-knot forward flight conditions are mildly unstable. The unstable poles represent a phugoid-like response in the front/side velocity coupled with the pitch and roll responses. In addition, there are modes that are at or near the origin corresponding either to a pure integration (for instance, the yaw variable derived from yaw rate) and to a lightly damped motion in the roll axis. Sensor outputs of the model consist of the body angular rates  $p$ ,  $q$ , and  $r$ , the body velocities  $u$ ,  $v$ , and  $w$ , the attitude angles roll  $\phi$ , pitch  $\theta$ , and yaw  $\psi$ , and the inertial positions  $x$ ,  $y$ , and  $z$ .

In both flight conditions, the linear models have a pair of defective degenerate eigenvalues at the origin. In forward flight, the yaw angle couples into the lateral  $y_{lat}$  displacement by way of forward velocity with  $\dot{y}_{lat}(t) = V_o\psi(t)$  included as part of the system model equations. (This is also true in the hover flight condition where there is a small, but not insignificant forward velocity). These degeneracies within the system matrix and the tendency to produce overlapping modes within the CLTR design procedure are major concerns in the numerical solution of the design optimization. These concerns are addressed directly with the robust algorithm presented in Section 3.3.

The four control actuator inputs are  $\delta_0$ ,  $\delta_s$ ,  $\delta_c$ , and  $\delta_{TR}$  denoting the collective (main rotor), the main rotor sine, the main rotor cosine, and the tail rotor pitch inputs respectively. Using these controls, one should be able to achieve a decoupled set of ideal command responses in heave, roll, pitch, and yaw rate.

### 7.3 Actuator and Sensor Delays

Only the actuator delays and the sensor delays for the command variables  $\theta$ ,  $\phi$ ,  $r$  and  $\dot{z}$  (Figure 7.2) are present in the state-feedback design. Both the actuator and all the sensor delays are accounted for in the output-feedback designs. These were set uniformly to 60 ms. Because of the sensor delays, the system has a set of nonminimum-phase zeros and it is impossible to recover state-feedback design properties with an output-feedback design.

This delay is modeled as a first-order Pade approximation model. In the Laplace domain, it is given by

$$T_{delay}(s) = \frac{1 - \frac{T_s}{2}s}{1 + \frac{T_s}{2}s}$$

or, in the time domain,

$$\begin{aligned}\dot{x}(t) &= -\frac{2}{T}x(t) + y_{in}(t) \\ y_{out}(t) &= \frac{4}{T}x(t) - y_{in}(t)\end{aligned}$$

A higher-order Pade approximation model for the time-delay could have been used; however, this would lead to a prohibitive number of delay states due to the large number of sensor outputs (a total of 12) in this design problem.

### 7.4 ADS-33C Requirements and the Ideal Response Model

Performance of a controller design can be evaluated based on a subset of tests defined in the document ADS-33C [35]. This subset of requirements is applicable to rotorcraft models linearized about specific flight conditions, and falls into the categories of bandwidth, attitude quickness, cross coupling, and gust response.

Desired bandwidth properties of the closed-loop command responses are used to define low-order idealized response models for each of the command variables. The state-feedback design is derived from the minimization of the error between the actual rotorcraft responses and the ideal model responses. Such a procedure provides designers a systematic way of attaining adequate response in the respective channels without imposing excessively high bandwidth upon the others. Also, similar to CLTR, the need to adjust the criterion weights to optimize the design is not as severe as it would be with a direct design approach. The ideal model responses are associated with pitch, roll, yaw rate, and heave commands, and form the dynamics of the feedforward controller in the final design.

Bandwidth requirements for both the forward and the hover flight conditions are similar, hence one feedforward command model is used for both conditions—with the exception that for turn coordination, a command for yaw rate has to be included with any roll command in forward flight. Otherwise, all commands are completely decoupled in these ideal models.

Based on the ADS-33C bandwidth requirements in Figure 1(3.3) of [35], pitch, roll, and yaw rate responses are idealized as critically damped second-order responses, while

heave motion is patterned after a first-order response (refer to Table 4(3.3) in the ADS-33C document). A first-order approximation to a time delay of 0.1 s is included in each of the 4 channels. As a result, the ideal model has a total of 11 states. The natural frequency of the pitch response is 3 rad/s, that of the roll response is 5 rad/s, and the yaw rate response has a bandwidth of 4 rad/s. Damping of the critically damped second-order responses is perturbed slightly away from 1 to avoid degenerate eigenvalues in these uncontrollable modes. These modes can cause numerical difficulties in the eigenvector decomposition method of the LQ synthesis.

According to Figure 2(3.3) of ADS-33C, the command bandwidth is determined from the frequency response of the transfer function between the command input and the commanded output where the phase crosses 135 degrees. Targeted command bandwidths are set at 4.33 rad/s for pitch response, 6.08 rad/s for roll response, and 5.27 rad/s for yaw rate response. Associated with each is a phase delay of 0.06s for pitch, 0.055s for roll, and 0.057s for yaw rate. While these are all Level I characteristics, the bandwidth is not excessively high. The heave response is set to a time constant of 3 s and a delay of 0.1s, which correspond to moderate Level I characteristics. These ideal responses are shown in Figure 7.6.

Simplicity of the control synthesis based on ideal model matching allows designers to address the bandwidth problem as well as other design considerations. One area of concern is the attitude quickness. It is defined as the ratio between the peak rate and the angular change in an attitude step command, and is an evaluation criterion quantified in Figure 4(3.3) of the requirements [35]. These requirements are subject to interpretation when testing with a linear model. Because the criterion curve for ratio of maximum rate to attitude change is strongly dependent on the attitude command level, the most stringent case (at a 5° commanded change) is considered. To satisfy these criteria, ideal models with a pitch ratio of 1.1, and one with a roll ratio of 1.8 have been selected that correspond to average Level II responses in the target acquisition and tracking mode. To further improve these responses, one would have to increase the bandwidth further, and this would place a too stringent requirement upon any state or output-feedback designs in terms of control bandwidth. It is further anticipated that these ratios would be accentuated in a more realistic nonlinear simulation test, hence they are kept unchanged at the selected Level II characteristics.

In the ADS-33C requirements described in items (3.3.9.2) and (3.4.4.2), pitch to roll cross coupling (and vice versa) is defined as the peak response in one attitude variable due to a step command in the other. Additional cross coupling tests in forward flight involve measuring the peak pitch response normalized by vertical acceleration in a step heave command (item 3.4.4.1.1), and the peak sideslip to a 1°-step command in roll (item 3.4.6.2). Additional hover condition tests examine the peak response and envelope of oscillatory response of yaw rate from a step command in heave (item 3.3.9.1).

Roll damping in forward flight is tested by measuring the envelope of the oscillatory roll response due to the roll command (item 3.4.6.1). While neither pitch nor yaw damping are directly specified, the damping constants for the pitch, roll, and yaw rate responses are all set to near 1 in the ideal models.

Additional hover condition requirements involve yaw and yaw rate responses to gusts (item 3.3.7.1). Tests for yaw responses to gusts are somewhat involved. In one test, one applies a steady 25-knot headwind, followed by a sudden 10-knot side gust. In the other test, the wind inputs are applied in the reverse order: a steady 25-knot side wind and then a sudden 10-knot head gust.

## 7.5 State Feedback Designs

With the idealized output responses defined in the ideal models, the design procedure for the state-feedback designs follows somewhat the concept of loop transfer recovery—on the commanded responses. No reasonable ideal model appears to exist for the gust responses. By minimizing the difference in the commanded responses, the use of a pole attractor formulation for desired closed-loop stability is no longer needed. A further advantage is the anticipation factor introduced in the synthesis through the inclusion of the actuator delay in the design. A block diagram of the design set-up is given in Figure 7.1.

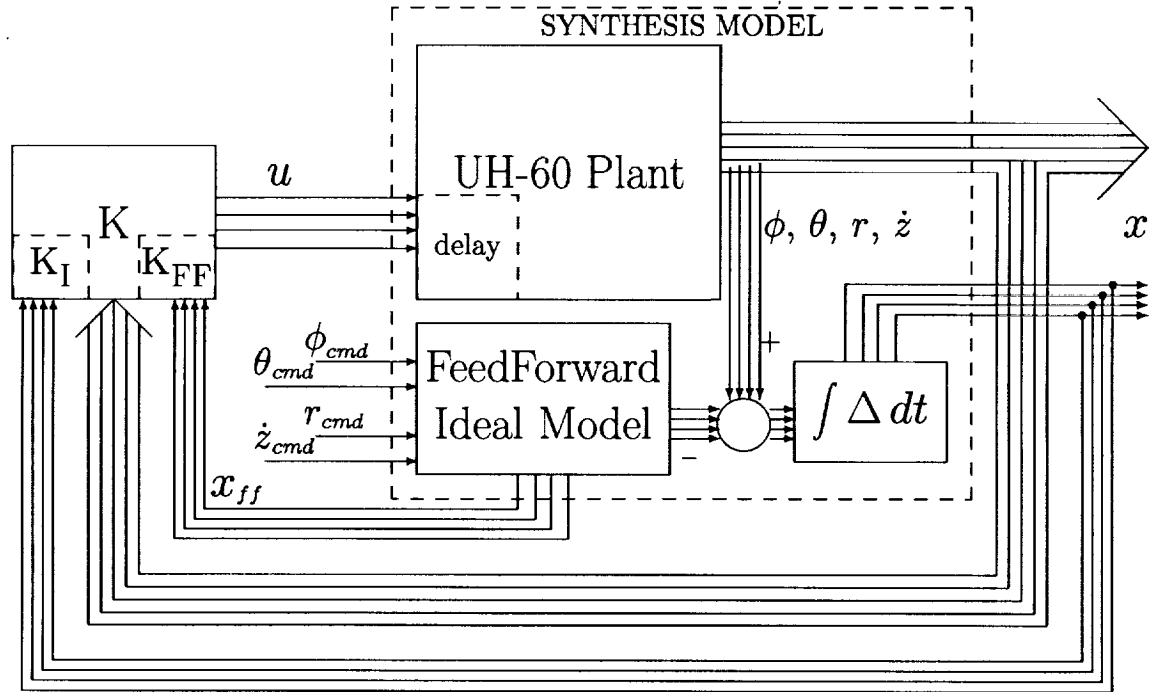


Figure 7.1: Synthesis Model for Full-State Design with Integral Controls

From our previous work [34], we recognize early on the need for integral control on the commanded error quantities. The purpose is to provide steady-state command tracking to step changes and incorporate design robustness to constant disturbances. Putting penalty weights on the integral of the commanded errors ensure that the integral modes are detectable from the LQ design cost function. The penalty weighting matrices  $Q$  and  $R$  differ significantly between designs for the hover condition and forward flight. The control penalty is evenly divided among the 4 controls with  $R = I$  (identity matrix). The criterion variables consist of  $\Delta\theta$  (pitch error),  $\Delta\phi$  (roll error),  $\Delta r$  (yaw rate error),  $\Delta\dot{z}$  (heave error), and integral of the errors  $\int\Delta\theta$ ,  $\int\Delta\phi$ ,  $\int\Delta r$ , and  $\int\Delta\dot{z}$ . In the hover condition, a criterion weight of 400 is used on all the variables with  $Q = 400I$ . In forward flight, however, a trade-off between performance and stability of an oscillatory mode around 18.5 rad/s is needed. This mode is excited primarily by the roll command. Sideslip control is of course a problem in forward flight, but an attempt to control it via a penalty on the sideways acceleration  $\dot{v}$  is not fruitful. Design effort involved in the selection of the penalty weights for the forward flight condition is considerably greater than that for the hover condition. The final weighting factors are given in Table 7.2. Robustness analysis in terms of single-loop gain and phase margins is also performed for both the control and sensor loops. Performance is

Table 7.2: Design Weights on the Criterion Variables in the LQ Synthesis

Criterion Variable	Hover	Forward Flight
$\int \Delta \theta$	400	500
$\int \Delta \phi$	400	500
$\int \Delta r$	400	80
$\int \Delta \dot{z}$	400	100
$\Delta \theta$	400	300
$\Delta \phi$	400	400
$\Delta r$	400	400
$\Delta \dot{z}$	400	500

satisfactory based on the ADS-33C requirements. Results are shown in Tables 7.7 to 7.10, and Figures 7.7, 7.8, 7.9, and 7.10.

Since there is no penalty weighting on  $\psi$ ,  $x$ ,  $y$ , or  $z$ , and that these states are associated with poles at the origin, they will not be detectable in the LQ synthesis. To avoid numerical difficulties in the solution of algebraic Riccati equation, these states have been removed from the state-feedback synthesis model. However, in the output-feedback designs, these additional sensor outputs would provide improved observability on the integrals of  $\dot{\psi}$ ,  $u$ ,  $v$ , and  $w$ . This information is especially helpful in steady-state command following for the variables  $r$  and  $\dot{z}$ .

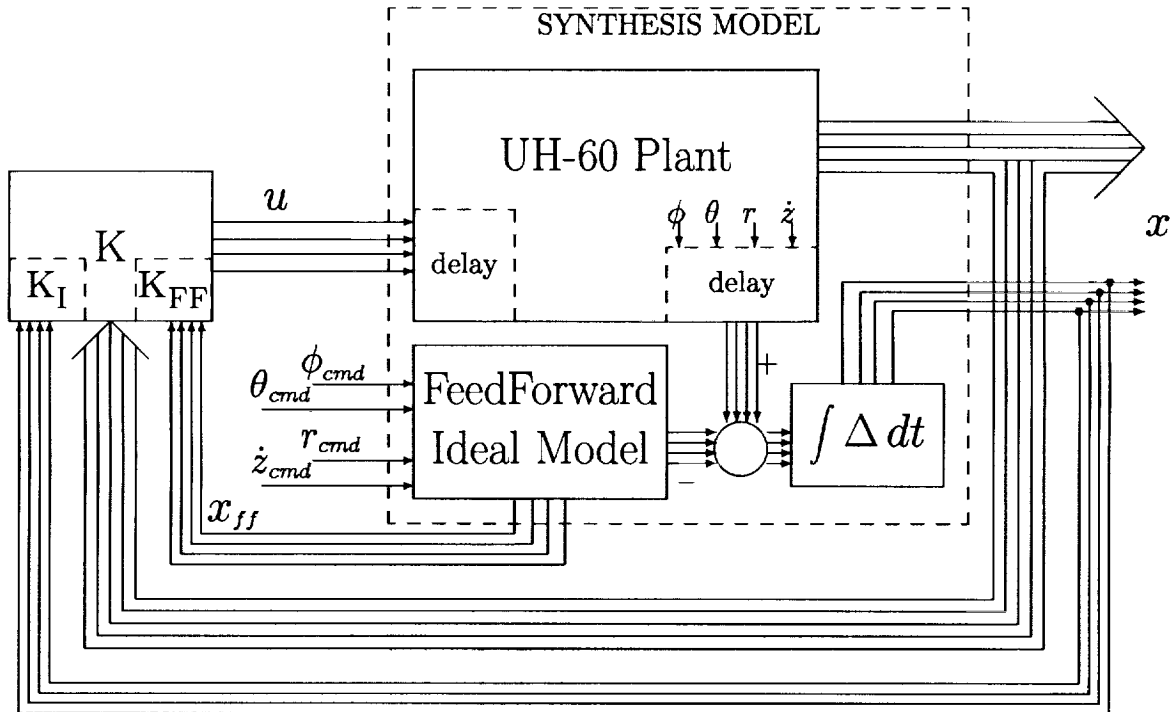


Figure 7.2: Synthesis Model for Full-State Design with Sensor Delays Added

With a state-feedback control structure as shown in Figure 7.1, all the state information is assumed available, uncorrupted by noises and not subject to sensor delay. However, for

proper control anticipation, sensor delays are taken into account in those sensor outputs that are used to form the integral errors. Hence, the state-feedback gains used in the subsequent CLTR observer-based designs are derived from the system shown in Figure 7.2. Performance of the resulting closed-loop state-feedback system defines the baseline results for comparison with subsequent output-feedback designs from loop transfer recovery or direct optimization. The integrators in the integral control of the command errors are clearly more a part of the feedback controller than of the plant model, and the state-feedback gains on the integral states can be retained in the reduced-order observer-based designs.

## 7.6 Closed-Loop Transfer Recovery

Closed-loop transfer recovery (CLTR) offers a useful framework for the synthesis of output-feedback controllers starting from a satisfactory state-feedback control laws. Analysis of closed-loop recoverability using the Special Coordinate Basis (SCB) transformation reveals that the system between the command/disturbance inputs and the measurement outputs has 23 internal states, 19 output states, 8 stable invariant zeros (at @33.33 rad/s), and 12 infinite zeros of order 1. Note that the system used in this analysis includes the sensor delay in the outputs to the integral control. In this SCB analysis, we include independent disturbance inputs to all the system states, with the exception of the output delay states (for pitch, roll, yaw rate, and heave), and the command error integral states. Again note that the states  $\psi$ ,  $x$ ,  $y$ , and  $z$  are not involved in the LQ synthesis and hence are excluded from the analysis.

Exact or asymptotic recovery is in general not possible. The existence of invariant zeros in the right half-plane due to time delay in both the inputs and outputs poses a constraint on the loop recoverability. However since these invariant zeros are far removed from the origin and hence will contribute only slightly to the non-recoverable error. The major difficult for loop recovery lies in the fact that the system is not left-invertible due to the presence of multiple disturbances entering into the majority of the system states.

It should also be noted that a full-order observer-based design generally achieves less overall recovery than a reduced-order (Luenberger) observer-based design since the latter design has the inherent benefit of feedthrough terms from the direct feedback of the measurement outputs.

## 7.7 Luenberger Design

From early experiences with similar rotorcraft designs, a Luenberger design tends to achieve better recovery of the closed-loop LQ performance. With direct observation of the integral states and the feedforward ideal model states, a Luenberger design is also more appropriate than a full-order observer-based controller. Furthermore, due to the large model size, one can make use of the sensor outputs to minimize the number of states that need to be included in the dynamics of the reduced-order observer. This is conveniently done by replacing parts of the system states with the noise-free measurement outputs.

Design of a Luenberger observer-based controller under the CLTR procedure is described in Sections 5.5 and 5.6. In terms of an  $H^2$ -norm minimization of the recovery error matrix, one usually needs to solve a singular optimal control problem. Generally, a reduction in the recovery error is accompanied by an increase in the magnitude of the observer gain matrix. Due to the presence of only infinite zeros of order 1, the recoverable portion can be exactly

recovered by a reduced-order Luenberger observer-based controller. A simple diagram of the CLTR design formulation can be seen in Figures 7.3 and 7.4.

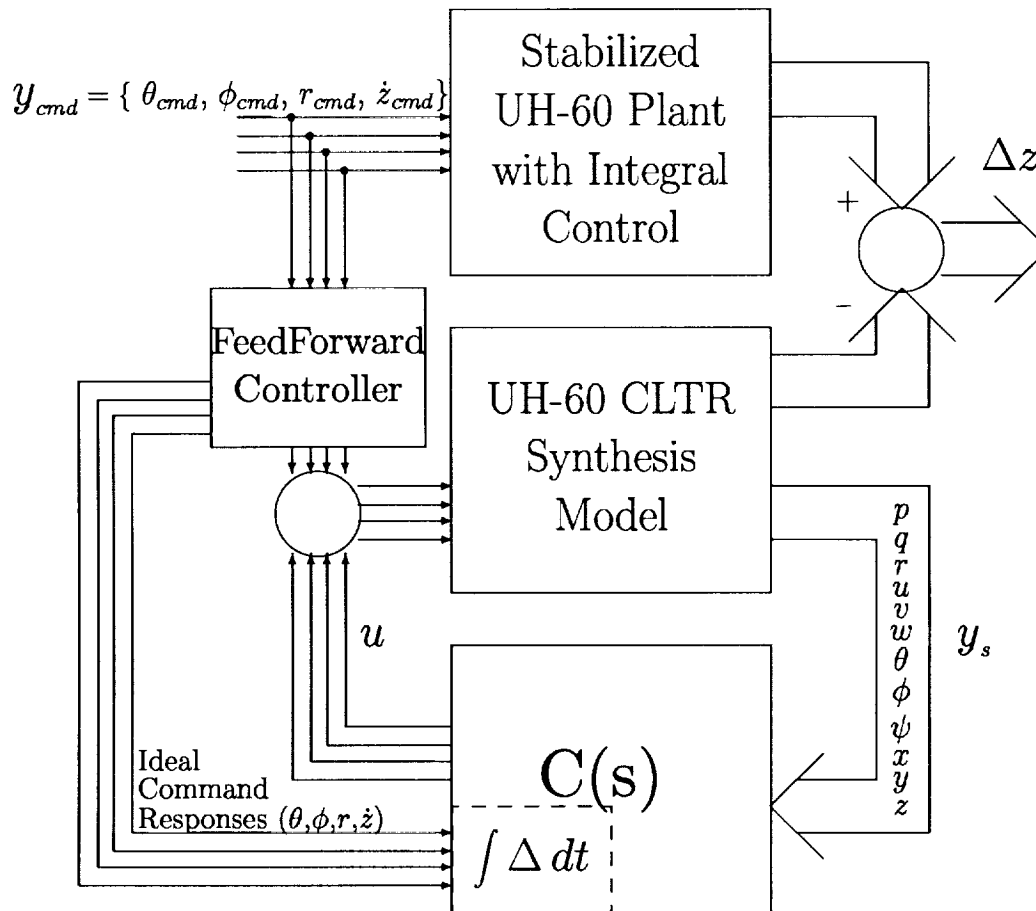


Figure 7.3: Block Diagram for CLTR with Output-Feedback Controller  $C(s)$

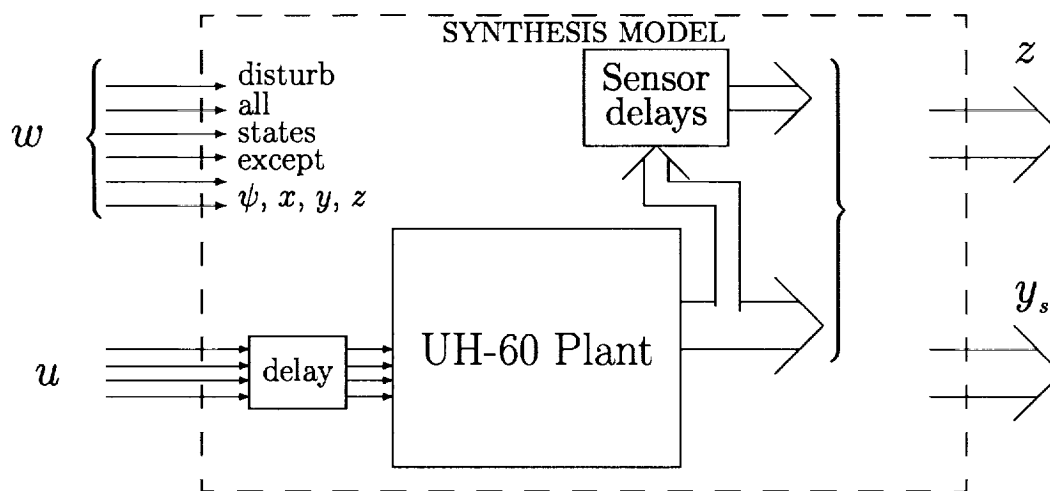


Figure 7.4: Synthesis Model for CLTR with Observer-Based Controllers

The overall results are satisfactory, with some degradation in command bandwidth and

relatively minor changes in performance (Tables 7.13 and 7.14). Robustness also remains acceptable (Tables 7.7 to 7.12). Some noticeable differences in the command responses are seen in Figures 7.7 to 7.10.

The overall controller design consists of a 35<sup>th</sup>-order Luenberger observer combined with an 11<sup>th</sup>-order feedforward controller, and an integral controller of 4<sup>th</sup> order. This compares to the 47<sup>th</sup>-order rotocraft model, which includes a 31<sup>st</sup>-order model for the rotocraft dynamics, a 4<sup>th</sup>-order model for the actuator delay, and a 12<sup>th</sup>-order model for the sensor output delay.

To describe more precisely the controller structure, let's designate the ideal model (i.e., feedforward controller) system as  $\{A_{ideal}, B_{ideal}, C_{ideal}, D_{ideal}\}$ , and the state matrices of the Luenberger design as

$A_{lc}$	Luenberger observer system matrix
$B_{lc}$	Control input distribution matrix
$C_{lc}$	Output distribution matrix (state to control)
$G_{lc}$	Sensor to observer input distribution matrix
$P_{lc}$	Sensor to control distribution matrix

Of particular interest are the  $G_{lc}$  and  $P_{lc}$  matrices. Note that the inputs to the controller include not only the usual sensor outputs  $y_s$  from the plant, but the integral states  $x_I$  and the feedforward controller states  $x_{ff}$  as well. One can then partition the matrix  $G_{lc}$  into  $[G_{lc}^S, G_{lc}^I, G_{lc}^{FF}]$  where  $G_{lc}^I = G_{lc}^{FF} = 0$ . The same partitioning applies to the matrix  $P_{lc} = [P_{lc}^S, P_{lc}^I, P_{lc}^{FF}]$ . These matrices show up in the overall controller model as follows, depending on whether it is described in

- an observer-like form:

$$\begin{aligned} A_c &= \begin{bmatrix} f\Delta & -C_{ideal} & 0 \\ 0 & A_{ideal} & 0 \\ 0 & 0 & A_{lc} \end{bmatrix}, \quad B_c = \begin{bmatrix} -D_{ideal} & C_{IS} & 0 \\ B_{ideal} & 0 & 0 \\ 0 & G_{lc}^S & B_{lc} \end{bmatrix} \\ C_c &= [P_{lc}^I \quad P_{lc}^{FF} \quad C_{lc}], \quad D_c = [0 \quad P_{lc}^S \quad 0] \end{aligned} \quad (7.1)$$

with inputs  $y_{cmd}$ ,  $y_s$ , and  $u$ , or in

- an alternate non-observer form:

$$\begin{aligned} A_c &= \begin{bmatrix} f\Delta & -C_{ideal} & 0 \\ 0 & A_{ideal} & 0 \\ B_{lc}P_{lc}^I & B_{lc}P_{lc}^{FF} & A_{lc} + B_{lc}C_{lc} \end{bmatrix}, \quad B_c = \begin{bmatrix} -D_{ideal} & C_{IS} \\ B_{ideal} & 0 \\ 0 & G_{lc}^S + B_{lc}P_{lc}^S \end{bmatrix} \\ C_c &= [P_{lc}^I \quad P_{lc}^{FF} \quad C_{lc}], \quad D_c = [0 \quad P_{lc}^S] \end{aligned} \quad (7.2)$$

with no explicit input from the control  $u$ .

Note that the term  $C_{IS}$  distributes the sensor outputs  $\phi$ ,  $\theta$ ,  $r$ , and  $\dot{z}$  to the appropriate integral states. So far, for the purpose of robustness analysis, the observer-based form has been used. Proper selection of one form over the other would become important when we proceed to the reduction of the controller order.

## 7.8 Model Reduction on the Feedforward Controller

While the need to reduce the number of states in the 35<sup>th</sup>-order Luenberger observer design is evident, one could also reduce the order of the feedforward controller as well. Usually one tends to perform a reduction of the entire controller at once without regard to the internal structures of the resulting linear time-invariant controller. However, to apply the method of balanced truncation, one needs to partition out the integral states of the commanded errors. Furthermore, the dynamics of the feedforward controller are only coupled in one direction to the feedback portion of the controller, it will therefore be practical to do this reduction as a separate process.

Finally, one needs to examine the question as to whether one should reduce the components of the compensator in an observer-like form or in the alternate form. Putting the controller into the latter form will add a dependency to the feedforward controller states, as seen in the  $B_{lc}P_{lc}^{FF}$  term within the  $A_c$  matrix of equation 7.2. This term represents the distribution into a portion of the controller related to the Luenberger observer design, in the same place that  $G_{lc}^{FF}$  would have been. Output from this distribution would add more degrees of freedom to the model reduction. Even though there are more inputs, the observer-like form is actually simpler and therefore preferred.

Given a controller in observer-like form, the subsystem associated with the feedforward controller dynamics has inputs  $w_c = \{\theta_c, \phi_c, \psi_c, \dot{z}_c\}$ , and produces as outputs the ideal responses used in the feedforward commands  $w_{FF} = \{\theta_{FF}, \phi_{FF}, \psi_{FF}, \dot{z}_{FF}\}$ , along with the feedforward control  $u_{FF}$ . Hence, the resulting feedforward controller used for model reduction is represented by the following state matrices

$$A_{FF} = A_{ideal} \quad , \quad B_{FF} = B_{ideal}$$

$$C_{FF} = \begin{bmatrix} C_{ideal} \\ P_{lc}^{FF} \end{bmatrix} \quad , \quad D_{FF} = \begin{bmatrix} D_{ideal} \\ 0 \end{bmatrix}.$$

Remember that the output  $u_{FF}$  is associated with the distribution matrix  $P_{lc}^{FF}$ . In the model reduction, we seek the lowest model order that produces the closest input/output characteristics for the given command inputs. The preferred technique is based on the balanced truncation method. Looking at the Hankel singular values as listed in Table 7.3, one could identify that a 7<sup>th</sup>-order feedforward controller (i.e., a reduction of 4 states) appears satisfactory. The same conclusion applies to both flight conditions.

One problem with this reduction technique is that the static relation between  $y_{cmd}$  and  $w_{FF}$  in the resulting reduced-order feedforward controller is no longer unity. Usually curve fitting is used to further improve the results of the truncation procedure. In this case, we can make use of the additional degrees of freedom in the direct feedthrough term which is not modified in the balanced truncation procedure. The curve fit is done using numerical optimization. It is found that with white-noise inputs, the integral of the truncation error must be penalized in order for the steady-state values of the reduced-order model to match those of the original model. The error was reduced to less than a percent of its original value, and the resulting feedforward controller matches its full-order counterpart well.

In the feedforward ideal model, only diagonal terms are considered in the command input distribution matrix. In other words, there is no crossfeed from a pitch command to the ideal roll response. For the command-to-control portion, a full feedthrough matrix is allowed. The direct command-to-control submatrix  $D_{plc}^{ideal}$  is contained in the  $D_c$  matrix of

Table 7.3: Hankel Singular Values for Feedforward Controller Model

	Hover	15knot Forward
1:	13.4440	23.1376
2:	13.3388	13.6907
3:	12.7467	11.8742
4:	11.2347	6.9295
5:	2.5749	4.0842
6:	1.7669	1.5641
7:	1.5000	0.9651
8:	0.0654	0.1047
9:	0.0478	0.0476
10:	0.0438	0.0263
11:	0.0195	0.0202

the controller in observer-like form. Namely,

$$D_c = \begin{bmatrix} D_{Plc}^{ideal} & P_{lc}^{FF} & 0 \end{bmatrix}.$$

In the alternate form we have the same matrix in  $D_c$  plus an additional nonzero submatrix  $B_{lc}D_{Plc}^{ideal}$  within the controller matrix  $B_c$ ,

$$B_c = \begin{bmatrix} -D_{ideal} & C_{IS} \\ B_{ideal} & 0 \\ \underline{B_{lc}D_{Plc}^{ideal}} & G_{lc}^S + B_{lc}P_{lc}^S \end{bmatrix}$$

## 7.9 Balanced Truncation of Luenberger Observer

The observer part of the compensator has not only sensor inputs  $y_s$ , but also the control inputs  $u$  as well. These control inputs could be eliminated by separately closing the loop around the compensator, but this has two harmful side effects. Firstly, it makes impossible to identify and separate out in the compensator portions that correspond to the feedforward controller, the integral control, and the observer. Secondly, the open-loop compensator may possess some unstable poles, making the balanced realization impossible. Thus, it is essential to retain a separate input from the control  $u$  in what one would call an observer-like form of the compensator.

The observer portion of the controller model has the following state model description,

$$\dot{x}_o = A_o x_o + B_o \begin{bmatrix} y_s \\ u \end{bmatrix}$$

$$u' = C_o x_o + D_o u$$

where

$$A_o = A_{lc} \text{ , } B_o = \begin{bmatrix} G_{lc}^S & B_{lc} \end{bmatrix} \text{ , } C_o = C_{lc} \text{ , } D_o = \begin{bmatrix} P_{lc}^S & 0 \end{bmatrix}.$$

Examining the Hankel singular values of such a system (Table 7.6), we see that the model can possibly be reduced to a 25<sup>th</sup>-order controller without noticable change in performance.

Design analysis results confirm the fact that this truncated controller model performs almost as well as the full-order observer-based controller design. In most areas of robustness,

command response, and ADS-33C design requirements the reduced order design shows little difference from the full-order design.

Combined with the reduced-order feedforward controller (7 states) and the integral controller (4 states), the overall controller is reduced to 36<sup>th</sup> order.

Closer examination of the Hankel singular values reveals that further reduction in the order of the observer part of the controller might be possible leading to a 14<sup>th</sup>-order observer. Analysis of the controller with a 14<sup>th</sup>-order observer indicates that the design is not satisfactory, although the resulting closed-loop system remains stable. In the next section, this reduced-order observer will be redesigned using numerical optimization for improved performance and robustness.

## 7.10 Reduction of Controller Order using Numerical Optimization

Note that the feedforward controller order has been reduced separately, and the integral gains are left unchanged, the remaining task in controller order reduction involves only the observer portion of the controller. As such, only this portion of the controller is formulated for optimization using the design tool SANDY.

The controller state matrix  $A_c$  matrix is allowed to have a tridiagonal structure which was derived from model reduction of the Luenberger observer. All of the elements in the matrix  $A_c$  identified by the asterisk are design variables in the optimization.

$$A_c = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

Furthermore, elements of the remaining controller state matrices  $B_c$ ,  $C_c$ , and  $D_c$  are also chosen as design parameters.

$$\begin{aligned} B_c &= \left[ \begin{array}{c|c} G_{lc} & B_{lc} \\ \hline (12 \text{ sensor inputs}) & (4 \text{ control inputs}) \end{array} \right] \\ C_c &= (4 \text{ control outputs}) [C_{lc}] \\ D_c &= (4 \text{ control outputs}) \left[ \begin{array}{c|c} P_{lc} & 0 \\ \hline (12 \text{ sensor inputs}) & (4 \text{ control inputs}) \end{array} \right] \end{aligned}$$

While the 14<sup>th</sup>-order observer-based controller arrived at by balanced truncation is inadequate, it can however be used as a starting point from which one can recover the full-order

controller responses through numerical optimization. We define the design objective function in the design algorithm SANDY to be the  $H^2$ -norm of the error in transfer function between the original Luenberger observer and the above specified 14<sup>th</sup>-order design. As in the optimization of the feedforward controller, it would be desirable to incorporate more design degrees of freedom into the lower-order observer structure without increasing the overall controller order. However, because the optimization already involves 368 variables, introducing additional degrees of freedom will have to be done carefully. One could for example allow a direct feedthrough term from the control input  $u$  to itself, but it is found to be unnecessary for a well-posed system. One could also have introduced a direct link from the dynamics of the feedforward controller and the integral control to the observer dynamics. However, this path has been partially fulfilled by the control input into the observer and the formulation would be redundant.

In consideration of the alternate formulation to the observer-like form, it is found that closing the control loop about the compensator is not beneficial. While it could be argued that eliminating the explicit control input  $u$  will reduce the parameter count, a closer examination indicates that there is no significant advantage.

Simultaneous optimization of parameters in both the feedforward controller and the Luenberger observer controller is more costly in terms of memory and computational time. Optimization of just the observer-based controller alone involves 368 parameters. Because the feedforward controller matches its full-order counterpart well, and because the results from separate optimization of the observer-based controller are satisfactory, the procedure involving simultaneous optimization is therefore not needed.

The resulting design optimization improves significantly the controller performance. While the performance is close to the other output-feedback controllers (i.e., small relative to the difference between the other output-feedback controllers and the state feedback controller), its robustness while still adequate is somewhat degraded.

## 7.11 Numerical CLTR

In general, one can also approach the closed-loop transfer recovery via direct numerical optimization. It should be emphasized that numerical CLTR is different than the procedure described in Section 7.10 related to controller order reduction. Here, one can theoretically start from any preferred size of the controller—the order selected is an arbitrary choice. The design objective is to determine the controller parameters by matching the closed-loop responses of the resulting output-feedback controller to those achieved under the state feedback for a given set of disturbance/command inputs.

In the design optimization, the 25<sup>th</sup>-order controller (derived from the 14<sup>th</sup>-order observer given in the previous section) provides a convenient starting place in this case due to its size, and its potential for further improvement.

Initially, the input excitations are bandwidth-limited to 25 rad/s and fed into both the command and gust input channels. The resulting controller design exhibits poor robustness in the control paths. Additional noises bandwidth-limited to 150 rad/s are then introduced into the control actuators (the high-bandwidth value is chosen to be above all the system modes). For the hover case, the intensity of the actuator noises is set at 0.666 compared to a unit intensity in all other input excitations. However, the intensity is increased to 1.0 for the forward flight case (The results are found to be relatively insensitive to the chosen value).

Included in the performance index are the errors in the command responses in roll, pitch, yaw rate, heave, and their integrals. For robustness, it is necessary to also include the difference in control responses in the performance index. Several design iterations are

Table 7.4: Objective Weights in Numerical CLTR

Criterion Variable	Hover	Forward Flight
Roll Residual $\Delta\phi$	400	300
Pitch Residual $\Delta\theta$	400	400
Yaw Rate Residual $\Delta r$	400	400
Heave Residual $\Delta\dot{z}$	800	500
$\int \Delta\phi$	700	500
$\int \Delta\theta$	400	500
$\int \Delta r$	400	80
$\int \Delta\dot{z}$	100	100
Collective Residual $\Delta\delta_0$	1	1
Main Rotor Sine Residual $\Delta\delta_s$	1	1
Main Rotor Cosine Residual $\Delta\delta_c$	1	1
Tail Rotor Residual $\Delta\delta_{TR}$	1	1

usually needed in the set up and selection of the weightings in this strongly non-recoverable case. Table 7.4 lists the weightings used—they are similar to, and based on the state-feedback performance objective. The optimization results show that no further improvement can be made to the design obtained in Section 7.10.

## 7.12 Direct Optimization

In this section, we examine design results obtained from the direct optimization of an objective function instead of applying the CLTR procedure. Here, the controller is designed by minimizing the difference between the ideal command responses and the actual responses along with terms involving control penalty. Initial values of the criterion weights are taken from the respective  $Q$  and  $R$  matrices defined in the LQ synthesis. The controller structure is chosen to be the same as that in the last section to further explore the potential for optimization beyond the current designs having the same order and structure. The controller order is found to be reasonably small for design implementation. The numerical CLTR results of Section 7.10 are used to provide initial design guess in the direct optimization.

As with the numerical CLTR, the system is excited by unit amplitude command and gust inputs as well as by fictitious actuator noises. The command and gust inputs are modelled as outputs of first-order filters with a bandwidth of 25 rad/s excited by white noises of unit intensity. The actuator noises are derived from first-order filters with a bandwidth of 150 rad/s. While the actuator noises are kept at unit amplitude in the forward flight case, they are adjusted somewhat in the hover case: 0.2 for the collective control, and 0.5666 for the other control actuators.

In the hover case, considerable improvement can be achieved. The weighting matrices are initially set to the values defined in the LQ synthesis, and undergo some adjustment through several design iterations. The final values can be seen in Table 7.5. A slight change

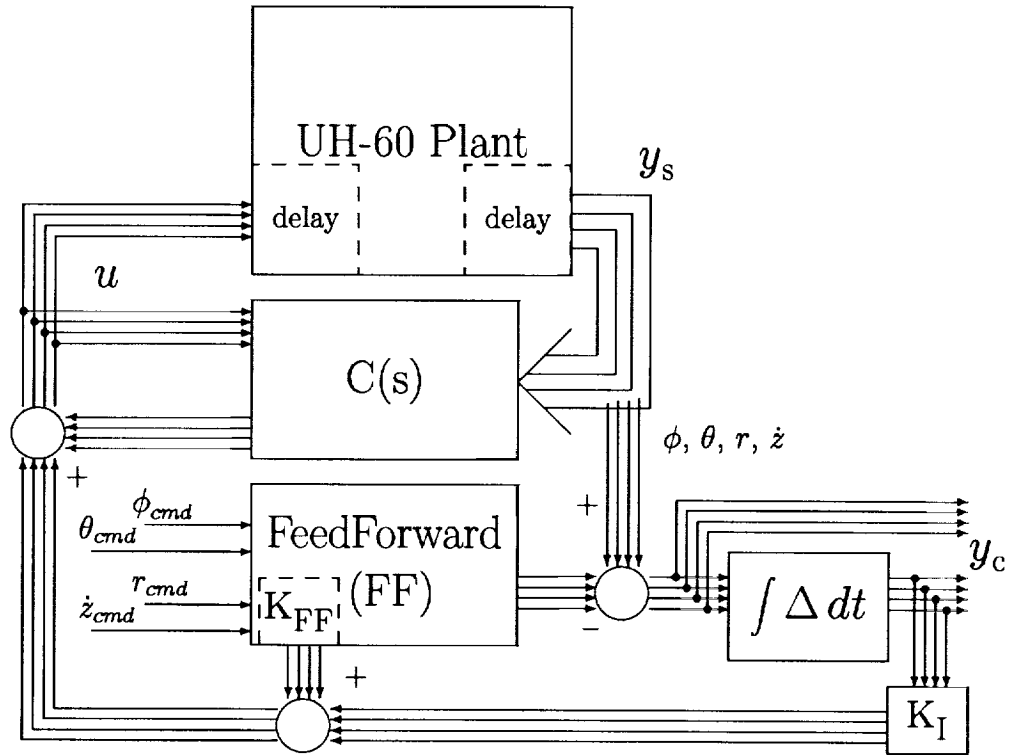


Figure 7.5: Controller Design Structure in Direct Optimization

Table 7.5: Objective Weights in Direct Optimization

Design Variable	Hover	Forward Flight
Roll Residual $\Delta\phi$	500	300
Pitch Residual $\Delta\theta$	500	400
Yaw Rate Residual $\Delta r$	400	400
Heave Residual $\Delta z$	800	500
$\int \Delta\phi$	750	500
$\int \Delta\theta$	650	500
$\int \Delta r$	400	80
$\int \Delta z$	300	100
Collective Control $\delta_0$	1	1
Main Rotor Sine Control $\delta_s$	1	1
Main Rotor Cosine Control $\delta_c$	1	1
Tail Rotor Control $\delta_{TR}$	1	1

in the forward flight design is found, but the difference is so slight that there is no noticeable changes in the design results. It would be difficult to characterize the difference in results for the forward flight, except that direct optimization may offer a slight edge in the overall robustness. Most of the improvement in the hover design is in the roll bandwidth (see Table 7.7). The improvement is gained at the expense of sensor robustness (Table 7.13).

## 7.13 Conclusions

Applicability of the CLTR technique has been demonstrated on the control of the UH-60 rotorcraft for two flight conditions. The results are presented for the hover and 15-knot forward flight conditions. They are found to be satisfactory in terms of ADS-33C requirements. It turns out that the feedforward controller dynamics at the two flight conditions are nearly the same, but a low-order set of ideal response dynamics should be feasible for any flight condition. With the given ideal model dynamics, a reasonable state-feedback design can be synthesized.

Not all the requirements in ADS-33C can be tested in the regime of linear models. Generally the controllers are found to meet the requirements for Level I qualities, except, as noted, for pitch and roll rate responses. The 14<sup>th</sup>-order Luenberger observer-based controller for hover does not meet Level I characteristics for roll bandwidth, and the result has been improved via direct optimization. This is not indicative of the limitations in the CLTR design procedure. It is believed that an improved state-feedback design for the hover case would have led to better output-feedback controllers via CLTR.

The proposed design technique based on CLTR should be applicable to other flight conditions. Evaluation of the CLTR design process for the hover and the forward (15 knot) flight conditions shows that most of the design tradeoffs occur in the synthesis of the state-feedback design. Once the problems of stability, performance and robustness have been solved in the LQ synthesis, recovery of these closed-loop properties becomes a much more routine process under CLTR. Although the procedure of numerical CLTR requires the tailoring of objective weighting matrices for an optimum design at each flight condition, an accurate selection of the weighting matrices is not significant to the overall results.

The order of the overall synthesis model in a CLTR design using numerical optimization is nearly prohibitive. Some shortcuts are performed to reduce the size of the overall CLTR system. A reduced-order (7 state) feedforward controller is used to drive both the state feedback (i.e., target) closed-loop model and the nominal open-loop model. The state-feedback system with 4 sensor output delay states is 35<sup>th</sup> order, while the open-loop plant with a full set of sensor output delays is 47<sup>th</sup> order. With integral control (4 states) for both state- and output-feedback, and a 14<sup>th</sup>-order observer, the overall dynamics quickly add up to a synthesis model of 111 states. An optimization on such a system takes substantial computing time, and is prone to have defective eigenvalues throughout the optimization run. The design results rely heavily on the reliable algorithm developed in Chapter 3.

The procedure in direct optimization design is facilitated considerably by the results from CLTR. The initial choice of objective weights can be developed from the LQ synthesis, and the initial controller can be taken from the CLTR design. In fact, the choice of controller order would not have been possible without the CLTR design followed by a model reduction procedure. One can conclude that design based on direct optimization would have entailed a much more tedious process if not for the useful insights developed from the CLTR procedure. The CLTR process becomes truly an integral part of the direct optimization design.

Table 7.6: Hankel Singular Values, Luenberger Observer

	Hover	15knot Forward
1:	19.6542	22.3400
2:	16.1352	18.2805
3:	14.3219	15.3796
4:	11.1515	10.5320
5:	9.9974	9.0218
6:	6.5465	7.0539
7:	4.7171	6.7871
8:	3.4355	4.8661
9:	3.2165	3.3467
10:	2.7132	3.2328
11:	2.1561	2.4144
12:	1.8829	1.8143
13:	1.6118	1.2724
14:	1.1177	0.7836
15:	0.5927	0.6148
16:	0.4878	0.4694
17:	0.3356	0.3540
18:	0.2910	0.2724
19:	0.2061	0.1688
20:	0.1795	0.1533
21:	0.1721	0.1211
22:	0.1270	0.1111
23:	0.1071	0.0884
24:	0.0967	0.0551
25:	0.0190	0.0495
26:	0.0003	0.0030
27:	0.0002	0.0025
28:	0.0000	0.0005
29:	0.0000	0.0000
30:	0.0000	0.0000
31:	0.0000	0.0000
32:	0.0000	0.0000
33:	0.0000	0.0000
34:	0.0000	0.0000
35:	0.0000	0.0000

Table 7.7: Single-Loop Sensor Robustness Properties, Hover

Design	Roll		Pitch		Heave		Yaw Rate	
	Gain	Phase	Gain	Phase	Gain	Phase	Gain	Phase
Full State +Integrals	6.13db, <-40db	52.36	8.84db, <-40db	48.85	+7.15db, <-40db	58.48	+5.97db, <-40db	57.39
Luenberger Observer	+5.64db, <-40db	49.92	+6.88db, <-40db	49.75	+5.01db, <-40db	54.80	+6.31db, -16.04db	44.75
25th Order Luenberger	+6.43db, <-40db	49.93	+6.60db, <-40db	49.81	+5.39db, <-40db	50.17	+6.73db, <-40db	44.77
14th Order Luenberger	+3.98db, -12.41db	45.00	+5.60db, <-40db	46.68	+5.44db, <-40db	52.98	+5.24db, -13.78db	43.27
14th Order Direct Opt.	+4.03db, -6.89db	34.95	+5.98db, <-40db	43.71	+6.29db, -28.96db	58.93	+5.07db, -29.20db	41.95

Table 7.8: Single-Loop Actuator Robustness Properties, Hover

Design	Collective		Sine		Cosine		Tail Rotor	
	Gain	Phase	Gain	Phase	Gain	Phase	Gain	Phase
Full State +Integrals	>40db, <-40db	65.00	>40db, <-40db	80.63	>40db, <-40db	70.80	>+40db, <-40db	91.55
Luenberger Observer	>40db, <-40db	65.00	>40db, <-40db	84.59	>40db, <-40db	73.97	>+40db, <-40db	91.55
25th Order Luenberger	>40db, <-40db	65.00	>40db, <-40db	80.63	>40db, <-40db	70.79	>+40db, <-40db	91.55
14th Order Luenberger	>40db, <-40db	48.40	>40db, <-40db	74.97	>40db, <-40db	59.89	>+40db, <-40db	97.21
14th Order Direct Opt.	>40db, <-40db	61.28	>40db, <-40db	57.46	>40db, <-40db	67.38	>+40db, <-40db	103.82

Table 7.9: Multiloop Actuator Robustness Properties, Hover

Design	Independent			$\underline{\sigma}(I + G(s))$	$\underline{\sigma}(I + G^{-1}(s))$
	+G.M.	-G.M.	P.M.		
Full State	$+\infty$ db,	-10db	60°	0.992	0.693
Luenberger Observer	$+\infty$ db,	-10db	60°	0.992	0.693
25th Order Luenberger	$+\infty$ db,	-10db	60°	0.992	0.693
14th Order Luenberger	+12,	-7	40	0.715	0.571
14th Order Direct Opt	+12,	-7	40	0.700	0.607

Table 7.10: Single-Loop Sensor Robustness Properties, Forward Flight

Design	Roll		Pitch		Heave		Yaw Rate	
	Gain	Phase	Gain	Phase	Gain	Phase	Gain	Phase
Full State +Integrals	6.86db, <-40db	49.89	8.48db, <-40db	47.70	+6.16db, <-40db	61.03	+7.24db, <-40db	61.20
Luenberger Observer	+5.49db, <-40db	49.70	+6.82db, <-40db	44.99	+6.89db, <-40db	61.03	+4.39db, <-40db	44.94
25th Order Luenberger	+5.38db, <-40db	49.36	+6.84db, <-40db	44.99	+6.89db, <-40db	64.80	+5.29db, <-40db	43.91
14th Order Luenberger	+5.41db, <-40db	44.70	+6.56db, -28.61db	43.88	+6.48db, <-40db	54.95	+5.43db, <-40db	42.01
14th Order Direct Opt.	+5.41db, <-40db	44.70	+6.85db, -28.61db	43.88	+6.48db, <-40db	54.95	+4.61db, <-40db	42.01

Table 7.11: Single Loop Actuator Robustness Properties, Forward Flight

Design	Collective		Sine		Cosine		Tail Rotor	
	Gain	Phase	Gain	Phase	Gain	Phase	Gain	Phase
Full State +Integrals	>40db, <-40db	64.29	>40db, <-40db	74.95	>40db, <-40db	74.90	>+40db, <-40db	90.34
Luenberger Observer	>40db, <-40db	64.95	>40db, <-40db	74.77	>40db, <-40db	75.00	>+40db, <-40db	90.34
25th Order Luenberger	>40db, <-40db	64.28	>40db, <-40db	74.84	>40db, <-40db	75.00	>+40db, <-40db	90.34
14th Order Luenberger	>40db, <-40db	45.93	>40db, <-40db	54.34	>40db, <-40db	57.23	>+40db, <-40db	90.44
14th Order Direct Opt.	>40db, <-40db	45.93	>40db, <-40db	54.94	>40db, <-40db	59.51	>+40db, <-40db	90.44

Table 7.12: Multiloop Actuator Robustness Properties, Forward Flight

Design	Independent			$\sigma(I + G(s))$	$\sigma(I + G^{-1}(s))$
	+G.M.	-G.M.	P.M.		
Full State	$+\infty$ db,	-8.5db	60°	1.000	0.632
Luenberger Observer	$+\infty$ db,	-8.5db	60°	1.000	0.632
25th Order Luenberger	$+\infty$ db,	-8.5db	60°	1.000	0.632
14th Order Luenberger	+12,	-9	40	0.710	0.660
14th Order Direct Opt	+12,	-9	40	0.709	0.659

Table 7.13: Design Evaluation, Hover

Test Result		ADS-33C Target	Full State Feedback	Luenberger Observer	25th Order Luenberger	14th Order Luenberger	14th Order Direct Opt.
Pitch	Bandwidth	Fig 1a,c (3.3)	3.344	2.642	2.648	2.699	2.510
	Phase Delay		0.147	0.187	0.186	0.184	0.182
Roll	Bandwidth	Fig 1b,d (3.3)	5.333	3.626	3.649	2.819	3.485
	Phase Delay		0.080	0.120	0.120	0.086	0.104
Yaw	Bandwidth	Fig 5 (3.3)	5.311	4.112	4.116	4.240	4.130
	Rate Phase Delay		0.061	0.104	0.105	0.102	0.113
Heave	Time Constant	< 5.0	2.981	2.872	2.856	2.816	2.876
	Delay (s)		< 0.2	0.169	0.173	0.171	0.195
Pitch	Rate from 1° step	> 1.6 < 0.25	1.160	1.177	1.175	1.234	1.220
	from Roll		0.012	0.010	0.010	0.016	0.007
Roll	Rate from 1° step	> 2.4 < 0.25	1.813	1.832	1.829	1.914	2.020
	from Pitch		0.009	0.011	0.011	0.022	0.009
Yaw	from side gust	< 0.66	0.264	0.292	0.292	0.297	0.283
	from head gust	< 0.66	0.264	0.292	0.292	0.297	0.283
	peak, from heave	< 0.65	5.8e-4	8.2e-4	9.1e-4	65.1e-4	59.7e-4
	oscil., from heave	< 0.2	5.8e-4	8.1e-4	11.8e-4	13.2e-4	45.7e-4
	for 1° Yaw change	> 2.4	2.800	2.773	2.835	2.861	3.430

Table 7.14: Design Evaluation, Forward Flight

Test Result		ADS-33C Target	Full State Feedback	Luenberger Observer	25th Order Luenberger	14th Order Luenberger	14th Order Direct Opt.
Pitch	Bandwidth	Fig 1 (3.4)	3.741	3.301	3.296	3.325	3.324
	Phase Delay		0.150	0.190	0.189	0.180	0.180
Roll	Bandwidth	Fig 2 (3.4)	5.779	4.825	4.810	4.844	4.844
	Phase Delay		0.176	0.136	0.136	0.115	0.115
Yaw	Bandwidth	Fig 8 (3.4)	5.329	4.059	4.063	4.244	4.244
	Rate Phase Delay		0.061	0.105	0.107	0.112	0.112
Heave	Time Const.	<5.0	2.957	2.960	2.952	2.836	2.836
	Delay (s)		<0.2	0.174	0.176	0.178	0.178
Pitch	from vertical accel.	< 1.0 <0.25	0.00779	0.00779	0.00867	0.00822	0.00822
	from Roll Cmd.		0.002	0.002	0.002	0.006	0.006
Roll	Rate from 1° step	> 2.4	1.860	1.847	1.839	1.846	1.846
	Osc. from 1° step	Fig 5(3.4) <0.25	0.00223	0.00217	0.00151	0.00172	0.00171
	from Pitch Cmd.		0.014	0.013	0.013	0.012	0.012
Sideslip	from 1° Roll step	Fig 6(3.4)	0.110	0.110	0.111	0.105	0.105

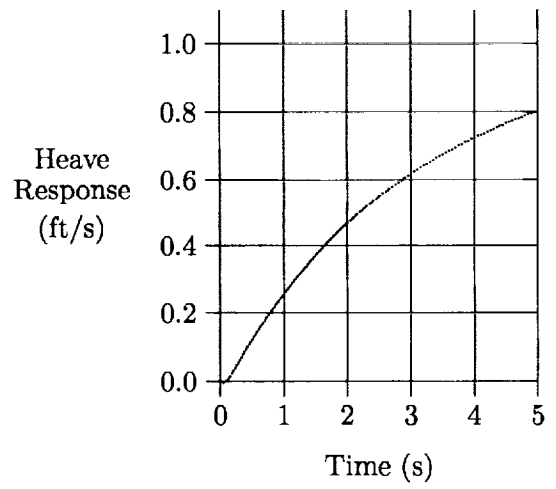
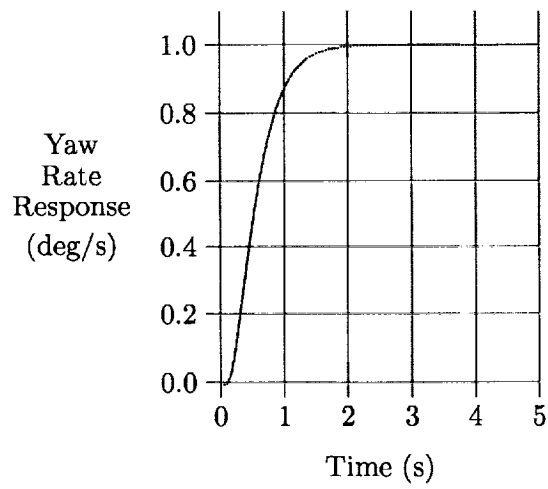
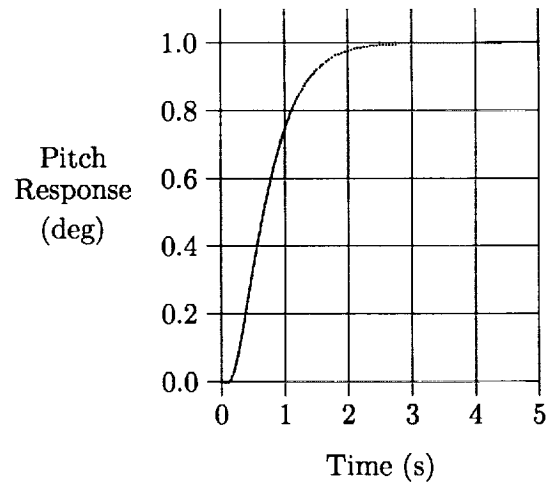
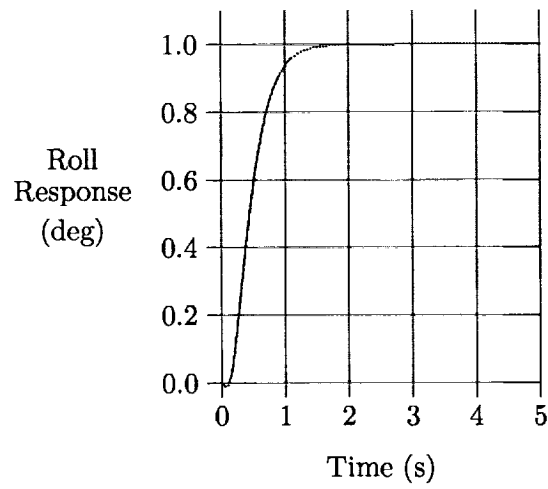


Figure 7.6: Ideal UH-60 Command Responses

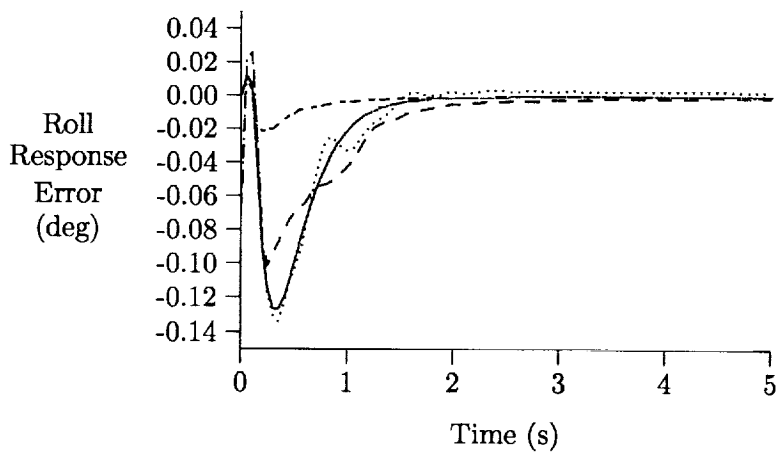
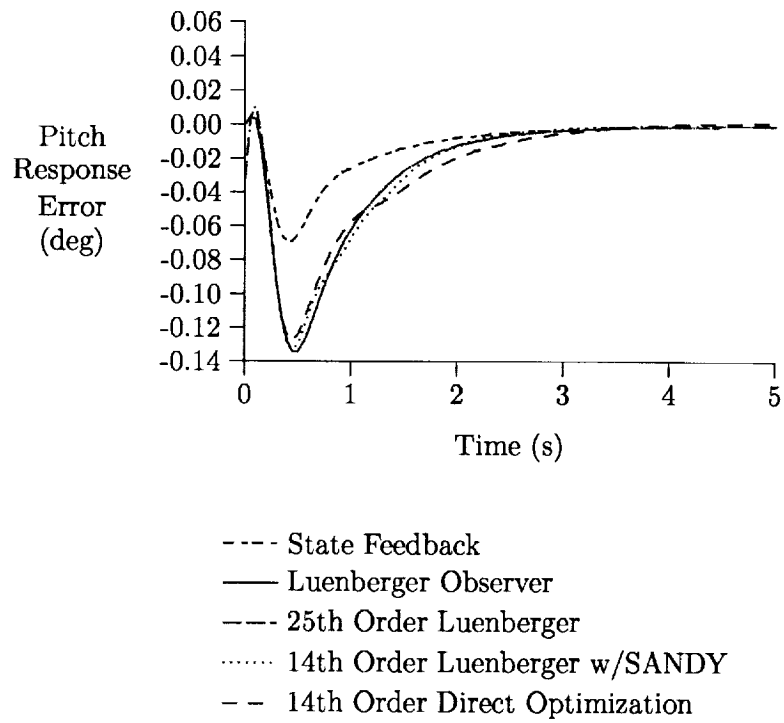


Figure 7.7: Errors from Ideal Responses, Hover

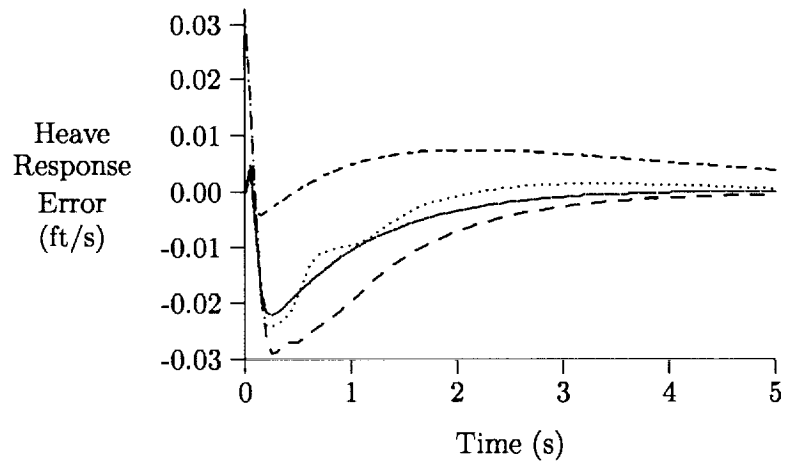
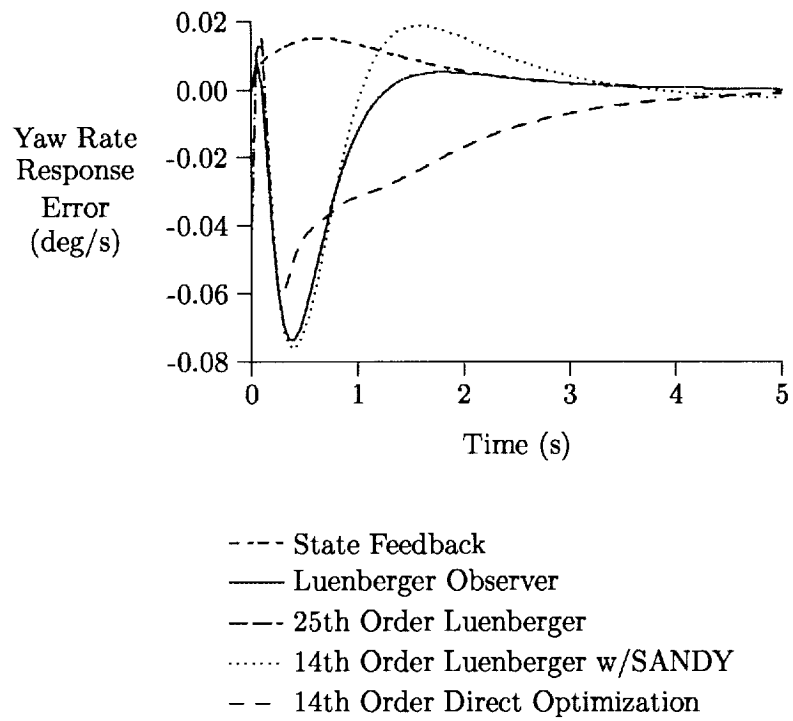


Figure 7.8: Errors from Ideal Responses, Hover

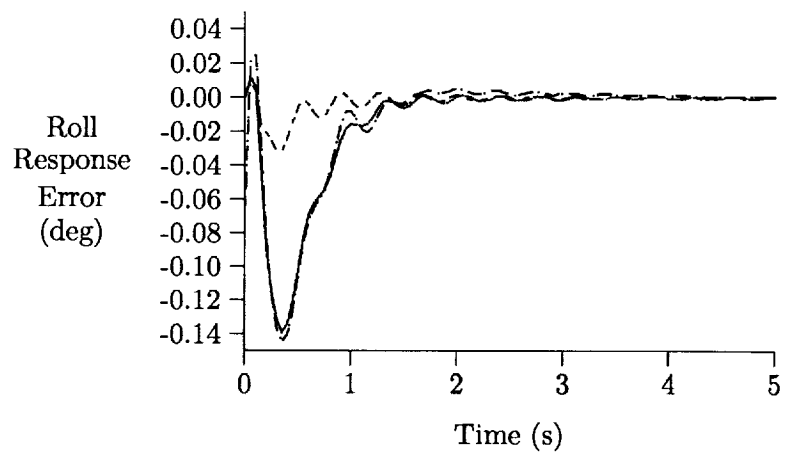
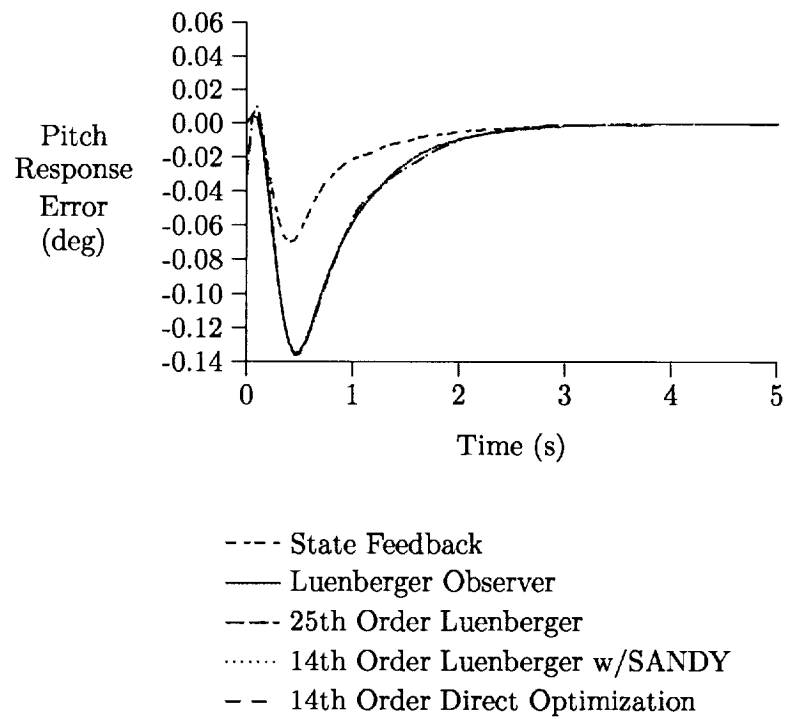


Figure 7.9: Errors from Ideal Responses, Forward Flight

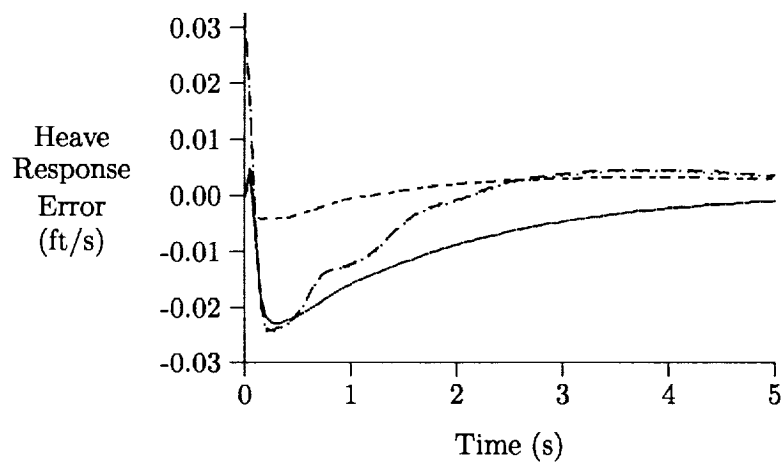
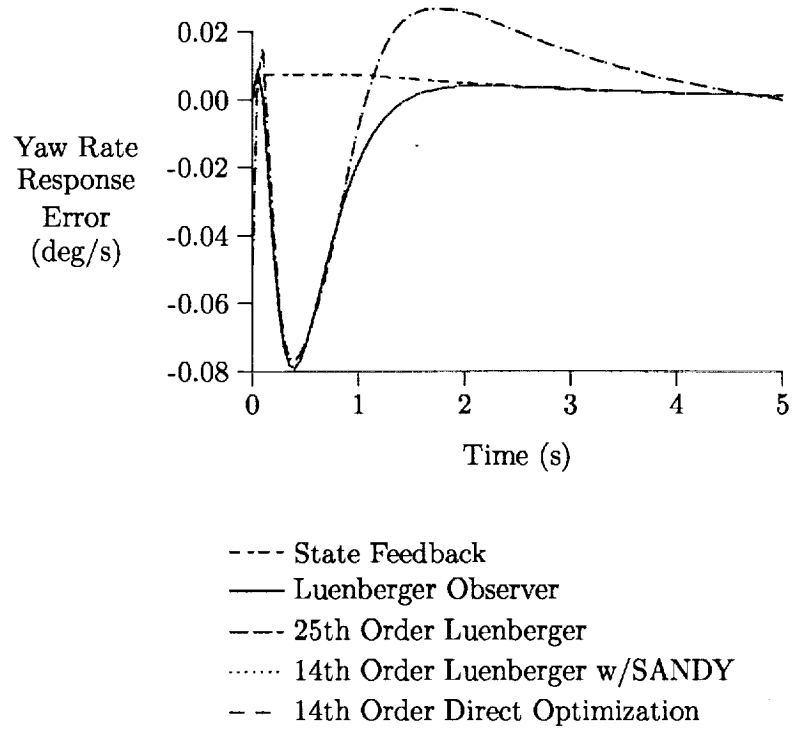


Figure 7.10: Errors from Ideal Responses, Forward Flight



## Chapter 8

# Future Work

### 8.1 Evaluation of Worst-Case Method for Loop Transfer Recovery

It is perceived that minimizing the worst-case difference between the closed-loop responses of a state feedback design and a those corresponding to a low order output-feedback controller case to a given range of disturbances would produce a more acceptable controller than from the  $H^2$  norm of the error. Given reasonable recovery, such a controller would be less prone to extreme variation from the state feedback characteristics. This would include loop transfer properties used in robustness measures. There exist several good test problems where this design algorithm can be exercised.

### 8.2 Completion of the Hybrid Algorithm

The decomposition of the system matrix into a partition of non-defective eigenvalues and a partition of defective eigenvalue blocks, as detailed in Appendix A, needs further evaluation. A thorough test must precede the conversion of the algorithms for computing  $\mathcal{X}$  and  $\mathcal{M}$  to the hybrid form.

The hybrid procedures for evaluating  $\mathcal{X}$  and  $\mathcal{M}$ , given in Appendix B, are more intricate than either the robust or diagonal forms due to the presence of cross terms. These cross terms are integrals containing both non-defective eigenvalues and defective eigenvalue blocks. The additional numerical complexity was relatively slight for the  $\mathcal{X}$  calculation where simple closed form solutions could be derived to reduce the complexity of the cross terms. Explicit handling of every cross term in a custom formula may yield rewards in speed and memory usage. However, in the case of  $\mathcal{M}$ , the problem is much more difficult. For example, for the double integral containing the defective eigenvalue block  $W_C$

$$\int_0^t \int_0^v e^{\lambda_i(v-s)} B_{i12} e^{W_C v} D_{21j} e^{\lambda_j s} ds dv,$$

it may still be as efficient to simply apply the robust algorithm using these arguments.

### 8.3 Eigenvalue/Damping Constraints

A tidy design is often achieved in pole placement. While exact pole placement is not possible within the context of numerical optimization, building a cost function based on

pole attractors [3] or forcing designated poles into a selected region [6] would seem to be. Creating a cost function that includes eigenvalue constraints and yet is robust to the degenerate mode condition is a subject for further research.

### 8.3.1 Current Method

The current method for eigenvalue constraint depends on the eigenvalue—eigenvector decomposition, and thus will not work when degenerate eigenvalues exist in the closed-loop system model.

The eigenvalue constraints and their gradients with respect to the controller design parameters are defined as follows. Given a system matrix  $A$  we let

$$Av_i = \lambda_i v_i \quad (8.1)$$

where  $\lambda_i$  and  $v_i$  are the eigenvalues and eigenvectors, respectively, of the system matrix  $A$ . We define a simple constraint function that is zero when the eigenvalues are in the desired stability region, or a nonzero value equal to the distance away from the desired stability boundary. One such function is

$$J_\sigma = \frac{1}{2} \sum_{i=1}^n \{ \max(\sigma_i - \sigma_{max}, 0) \}^2$$

This would reflect a cost function on real (or real part of) eigenvalues greater than a certain  $\sigma_{max}$ . It would be zero if all eigenvalues were less than  $\sigma_{max}$ .

In addition to constraining the real part, one usually needs to constrain the damping ratio

$$\zeta_i = -\frac{\sigma_i}{\sqrt{\sigma_i^2 + \omega_i^2}}$$

to be larger than a certain value (say  $\zeta_{min}$ ). Consider

$$J_\zeta = \frac{1}{2} \sum_{i=1}^n \{ \max(\sigma_i \sin \alpha + |\omega_i| \cos \alpha, 0) \}^2$$

where  $\cos \alpha = \zeta_{min}$  and  $\sin \alpha = \sqrt{1 - \zeta_{min}^2}$ . Each summand of this function will be zero if the corresponding eigenvalue satisfies the constraint; otherwise, it will be equal to the square of the distance from the offending eigenvalue to the damping line.

The above constraint functions are differentiable and their gradients are useful for numerical optimization, especially for the nonlinear programming algorithm NPSOL [21]. Given

$$J_\sigma = \frac{1}{2} \sum_{i=1}^n \{ \max(\sigma_i - \sigma_{max}, 0) \}^2$$

we have

$$\frac{\partial J_\sigma}{\partial p} = \sum_{i=1}^n \left\{ \max(\sigma_i - \sigma_{max}, 0) \frac{\partial \sigma_i}{\partial p} \right\}$$

Also, the damping constraint gradient is

$$\frac{\partial J_\zeta}{\partial p} = \sum_{i=1}^n \left\{ \max(\sigma_i \sin \alpha + |\omega_i| \cos \alpha, 0) \left( \frac{\partial \sigma_i}{\partial p} \sin \alpha + \text{sgn}(\omega_i) \frac{\partial \omega_i}{\partial p} \cos \alpha \right) \right\}$$

For non-degenerate eigenvalues, these gradients are well behaved.

Note that the constraint gradients are expressed in terms of the gradients of the closed-loop eigenvalues. Taking the derivative of equation (8.1) with respect to a parameter  $p$ , we have

$$\frac{\partial A}{\partial p} v_i + A \frac{\partial v_i}{\partial p} = \frac{\partial \lambda_i}{\partial p} v_i + \lambda_i \frac{\partial v_i}{\partial p}$$

or:

$$(A - \lambda_i I) \frac{\partial v_i}{\partial p} - \frac{\partial \lambda_i}{\partial p} v_i = -\frac{\partial A}{\partial p} v_i$$

The objective is to find an equation without a  $\frac{\partial v_i}{\partial p}$  term. One can diagonalize:  $A = T \Lambda T^{-1}$ , where  $\Lambda$  is a diagonal matrix containing the eigenvalues of  $A$ . With this transformation applied to the above equation, we have:

$$\begin{bmatrix} \lambda_1 - \lambda_i & 0 & & & & & 0 \\ 0 & & & & & & \\ & 0 & \lambda_{i-1} - \lambda_i & 0 & & & \\ & & 0 & 0 & 0 & & \\ & & & 0 & \lambda_{i+1} - \lambda_i & 0 & \\ & & & & & & 0 \\ 0 & & & & & 0 & \lambda_n - \lambda_i \end{bmatrix} T^{-1} \frac{\partial v_i}{\partial p} - T^{-1} v_i \frac{\partial \lambda_i}{\partial p} = -T^{-1} \frac{\partial A}{\partial p} v_i$$

The  $i^{th}$  equation from the above set shows a simple relation between  $\frac{\partial \lambda_i}{\partial p}$  and  $\frac{\partial A}{\partial p}$  without a term  $\frac{\partial v_i}{\partial p}$ , thus one can write:

$$\frac{\partial \lambda_i}{\partial p} = \frac{(T^{-1} \frac{\partial A}{\partial p} v_i)_i}{(T^{-1} v_i)_i}$$

The notation  $(\cdot)_i$  denotes the  $i^{th}$  row of the enclosed term. For complex eigenvalues,  $\sigma + i\omega$ , the real and imaginary parts of the above are  $\frac{\partial \sigma}{\partial p}$  and  $\frac{\partial \omega}{\partial p}$ , respectively. Note that when there is a near degeneracy and the eigenvalues are nearly defective,  $T^{-1}$  will be badly conditioned.

### 8.3.2 New Method of Eigenvalue Constraints

There are two equally important aspects of these constraints. First is the mechanization of the constraints themselves. The second is the ability to identify those parts of the system matrix where the constraints should apply. One often has disturbable yet uncontrollable modes where the eigenvalues do not obey the constraints and would upset the optimization process if so included. Moreover integral poles are often formulated to serve a control or estimation purpose.

It could be possible with the proposed method of decomposing the system matrix into eigenvalue blocks one could derive a means of calculating the eigenvalue derivatives for those blocks. All attempts so far have seemed preliminary.

Finally it may become that the only really promising method for forcing designated eigenvalues to a given region will be the plant transformation methods of Kawasaki and Shimemura [6] or Bernstein and Haddad [8].



# Bibliography

- [1] Stein, G. and Athans, M., "The LQG / LTR Procedure for Multivariable Feedback Control Design," *IEEE Transactions on Automatic Control*, AC-32, pp.105-114, 1987.
- [2] Brasch, F.M., and Pearson, J.B., "Pole Assignment Using Dynamic Compensation", *IEEE Transactions on Automatic Control*, Vol AC-15, No.1, 1970, pp. 32-43.
- [3] Gordon, V.C., and Collins, D.J., "Multi-Input Multi-Output Automatic Design Synthesis for Performance and Robustness", *Journal of Guidance and Control*, May-June 1986, Vol. 9, No. 3, pp. 281-287.
- [4] Doyle, J.C., Glover, K., Khargonekar, P.P. and Francis, B.A., "State-Space Solutions to Standard  $H_2$  and  $H_\infty$ -Control Problems," *IEEE Transactions on Automatic Control*, Vol. AC-34, No. 8, pp. 831-847, 1989.
- [5] Doyle, J.C. and Glover, K., "State-Space Formulae for All Stabilizing Controllers that Satisfy an  $H_\infty$ -Norm Bound and Relations to Risk Sensitivity," *Systems & Control Letters*, 11, pp. 167-172, 1988.
- [6] Kawasaki, N. and Shimemura, E., "A Method of Deciding Weighting Matrices in an LQ-Problem to Locate All Poles in the Specified Region", *IFAC Control Science and Technology (8th Annual World Congress) Kyoto, Japan, 1981*. pp 481-486.
- [7] Kimura, H., "Pole Assignment by Gain Output Feedback", *IEEE Transactions on Automatic Control*, Vol. AC-20, No.4, 1975, pp. 509-515.
- [8] Bernstein, D.S., and Haddad, W.M, "Controller Design With Regional Pole Constraints", *IEEE Transactions on Automatic Control* Vol 37, No.1, January 1992, pp 54-69.
- [9] Mills-Curran, W.C., "Calculation of Eigenvector Derivatives for Structures with Repeated Eigenvalues", *AIAA Journal*, Vol 27, No.7, July 1988, pp. 867-871.
- [10] Sannuti, P. and Saberi, A., "A Special Coordinate Basis of Multivariable Linear Systems-Finite and Infinite Zero Structure, Squaring Down, and Decoupling", *Int. J. Contr.*, Vol. 45, No. 5, pp. 1655-1704, 1987.
- [11] Chen, B.M., Saberi, A. and Ly, U., "Closed-loop transfer recovery with observer based controllers — Part 1: Analysis," *Proceedings of the 1991 AIAA GNC Conference*. Also, to appear in *Control and Dynamic Systems: Advances in Theory and Applications*.
- [12] Chen, B.M., Saberi, A. and Ly, U., "Closed-loop transfer recovery with observer based controllers — Part 1: Design," *Proceedings of the 1991 AIAA GNC Conference*. Also, to appear in *Control and Dynamic Systems: Advances in Theory and Applications*.

- [13] Stoorvogel, A.A., "The Singular  $H_\infty$ -Control Problem with Dynamic Measurement Feedback," *SIAM J. Control and Optimization*, Vol. 29, No. 1, pp. 160-184, 1991.
- [14] Stoorvogel, A.A. and Trentelman, H.L., "The Quadratic Matrix Inequality in Singular  $H_\infty$ -Control with State Feedback," *SIAM J. Control and Optimization*, Vol. 28, No. 5, pp.1190-1208, September 1990.
- [15] Levine, W.S. and Athans, M., "On the Determination of the Optimal Constant Output Feedback Gains for Linear Multivariable Systems," *IEEE Transactions on Automatic Control*, AC-15, pp.44-48, 1970.
- [16] Anderson, B.D.O. and Moore, J.B., *Linear Optimal Control*, Englewood Cliffs, Prentice Hall, Inc., New Jersey, 1971.
- [17] Makila, P.M., "Computational Methods for Parametric LQ Problems—A Survey," *IEEE Transactions on Automatic Control*, AC-32, No.8, pp.658-671, August 1987.
- [18] Ly, U., *A Design Algorithm for Robust Low-Order Controllers*, PhD. Thesis, Department of Aeronautics and Astronautics, Stanford University, November, 1982.
- [19] ElGhaoui, L., Carrier, A. and Bryson, A.E., "A New Performance Criterion for Worst Case Analysis and Design", *Journal of Guidance and Control*, July–August, 1992, pp. 953–961.
- [20] Mills, R.A., *Robust Controller and Estimator Design Using Minimax Methods*, PhD. Thesis, Department of Aeronautics and Astronautics, Stanford University, March 1992.
- [21] Gill, P.E., Murray, W., Saunders, M.A. and Wright, M.H., *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*. Technical Report SOL-86-2, Stanford university, January 1986.
- [22] VanLoan, C.F., "Nineteen Dubious Ways to Compute the Exponential of a Matrix", *SIAM Review* 20, pp.801-836, 1978.
- [23] Golub, G.H. and VanLoan, C.F., *Matrix Computations*. Johns Hopkins University Press, pp 396-400, 1985.
- [24] Bender, C.M. and Orzag, S.A. *Advanced Mathematical Methods for Scientists and Engineers*. McGraw-Hill 1978. pp 383-389.
- [25] Osborne, E.E. , "On Preconditioning of Matrices", *Journal of the Association for Computing Machinery*, Los Angeles, CA. pp 338-345. March, 1960.
- [26] Parlett, B.N. and Reinsch, C., "Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors", *Numerical Math*, 13, pp. 293-304 (1969).
- [27] Bavely, C.A. and Stewart, G.W., "An Algorithm for Computing Reducing Subspaces by Block Diagonalization". *SIAM J. Numerical Analysis*. Vol 16, No. 2, April 1979. pp 359-367.
- [28] Vivian, H.C. , Blaire, P.E., Eldred, D.B., Fleischer, G.E., Ih, C.-H.C., Nerheim, N.M., Scheid, R.E. and Wen, J.T., *Flexible Structure Control Laboratory Development and Technology Demonstration*, JPL Publication 88-29, October 1, 1987.

- [29] Westreich, D. , " A Practical Method for Computing the Exponential of a Matrix and its Integral", *Communications in Applied Numerical Methods*, Vol 6, pp. 375-380 (1990).
- [30] Wie, B. and Bernstein, D., "A Benchmark Problem for Robust Control Design," *Proceedings of the 1990 American Control Conference*, May 23-25, 1990.
- [31] Ge, Y., Collins, E.G., Watson, L.T., and Doris, L.D., "Minimal Parameter Homotopies for the  $L^2$  Optimal Model Order Reduction Problem", submitted for publication in *IEEE Transactions on Automatic Control*.
- [32] Kleinman, D.L., Fortmann, T., and Athans, M., "On the Design of Linear Systems with Piecewise-Constant Feedback Gains," *IEEE Transactions on Automatic Control*, Vol. AC-13, pp. 354-361, 1968.
- [33] Chen, B.M. , Saberi, A., Sannuti, P. and Shamash, Y., "Construction and Parameterization of All Static and Dynamic  $H^2$ -Optimal State Feedback Solutions," *Proceedings of the 31st IEEE Conference on Decision and Control*, December 1992. Also, to appear in *IEEE Transactions on Automatic Controls*, March 1993.
- [34] Heffley, R.K., "A Compilation and Analysis of Helicopter Handling Qualities Data", NASA Contractor Report 3145, Part II, August 1979.
- [35] "Aeronautical Design Standard" (ADS-33C), *Handling Qualities Requirements for Military Rotorcraft*, U.S. Army Aviation Systems Command Directorate for Engineering, St. Louis, MO, August 1989.
- [36] ElGhaoui, L., Carrier, A. and Bryson, A.E., *Linear Quadratic Minimax Controllers*, *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4 (July-August 1992), pp 954-961.
- [37] Khargonekar, P.P. and Rotea, M.A., "Mixed  $H_2/H_\infty$  Control: A Convex Optimization Approach," *IEEE Transactions on Automatic Control*, Vol. 36, No. 7, pp. 824-837.
- [38] Chen, B.M. , Saberi, A. and Ly, U. "Simultaneous  $H_2/H_\infty$  Optimal Control: The State-Feedback Case," *Proceedings of the 1992 AIAA Guidance, Navigation and Control Conference*, August 10-12, 1992.
- [39] Doyle, J.C., and Stein, G., "Robustness with Observers", *IEEE Transactions on Automatic Control*, AC-24, pp. 607-611, 1979.
- [40] Ly, U., " $H^2$ - and  $H^\infty$ -Design Tools for Linear Time Invariant Systems", *Proceedings of the 3rd Annual Conference on Aerospace Computational Control*, Volume I, Dec 15, 1989, JPL Pub. 89-45, Vol I.
- [41] Ly, U., VanSteenwyk, B., and Schömig, E., "Robust Control Design Using Parameter Optimization Techniques", *Control and Dynamic Systems*, C.T. Leondes, ed., Academic Press, 1993.



## Appendix A

# Preliminaries to the Hybrid Algorithm

### A.1 Introduction

It has been shown that the original gradient computation based on the diagonal form is fast, though inaccurate for defective systems. While the robust form described in Section 3.3 is accurate under most circumstances, it is also slower by an estimated factor of 8 or 9. The next question is whether one can apply the faster algorithm to the non-defective part of the matrix and the slower one to the defective part, thereby resulting in a fast *and* robust overall numerical scheme. Until such a scheme is actually implemented and tested, one can only speculate. There is a clear direction, however, as to how to structure that procedure.

In order to have control over the separation of the diagonalizable part and the defective part one must abandon the simple eigenvalue-eigenvector decomposition for a combination of the Schur decomposition and a procedure to separate the set of defective eigenvalues from the non-defective ones. This custom routine can then feed the actual hybrid computation algorithm with the appropriate parts of the input matrix.

Once a matrix is in Schur form, there will be a block diagonal (consisting of individual real eigenvectors and 2x2 blocks for complex pairs) along with an upper triangular part. The 2x2 blocks are not yet in  $\sigma - \omega$  form. Moving non-defective eigenvalues apart from the defective ones usually requires zeroing the upper-triangular interaction between these roots. In fact, the very definition of a defective root depends on the ability to zero this cross-term.

### A.2 Converting a 2x2 block to $\sigma - \omega$ form.

Conversion of a general 2x2 diagonal block to the form:  $\begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix}$  depends on a rotation and a scale conversion,

$$\begin{bmatrix} 1/d_1 & 0 \\ 0 & 1/d_2 \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a & b \\ f & g \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$$

where  $s = \sin(\phi)$  and  $c = \cos(\phi)$ . The result of the rotation is to bring the two diagonal elements to equality. The rotated matrix becomes

$$\begin{bmatrix} c^2a + sc(f+b) + s^2g & -s^2f + sc(g-a) + c^2b \\ c^2f + sc(g-a) - s^2b & s^2a - sc(f+b) + c^2g \end{bmatrix}.$$

Thus the condition on the diagonal elements results in

$$c^2a + sc(f+b) + s^2g = s^2a - sc(f+b) + c^2g \text{ or:}$$

$$2sc(f+b) = (c^2 - s^2)(g-a)$$

$$\sin(2\phi) = 2sc = \frac{(g-a)}{\Delta}$$

$$\cos(2\phi) = c^2 - s^2 = \frac{(f+b)}{\Delta} \text{ where}$$

$$\Delta = \sqrt{(f+b)^2 + (g-a)^2}$$

Given values for  $\sin(\phi)$  and  $\cos(\phi)$ , it remains to find the scaling terms that will bring the off diagonal terms to being the opposite of one another (since the scaling will not affect the diagonal terms). The result is:

$$\begin{aligned} \text{or} \quad \frac{d_2}{d_1} \left( \frac{(f-b)}{2} - \frac{\Delta}{2} \right) &= \frac{d_1}{d_2} \left( \frac{(f-b)}{2} + \frac{\Delta}{2} \right) \\ \frac{d_2}{d_1} &= \sqrt{\frac{\frac{(f-b)}{2} + \frac{\Delta}{2}}{\frac{(f-b)}{2} - \frac{\Delta}{2}}}. \end{aligned}$$

Of course the argument under the square root must be positive, which can be written as  $(f-b)^2 > \Delta^2$ .

### A.3 Reduction of a 2x2 Block to Upper Triangular Form

The transformation from an upper Hessenburg form to the real Schur form for a matrix depends on the QR iteration. This iteration has some difficulty for a matrix with defective degenerate eigenvalues. Thus, there can be nonzero sub-diagonal elements not associated with complex eigenvalues. It is possible for one to see nonzero subdiagonal elements for a whole Jordan block, but usually one sees only a scattering of 2x2 diagonal blocks. It is sometimes necessary to check each 2x2 block to see if it is non-converged or if it *really* is a complex pair. Zeroing a sub-diagonal element can be done with a Givens rotation

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a & b \\ f & g \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

resulting in

$$\begin{bmatrix} c^2a + sc(f+b) + s^2g & -s^2f + sc(g-a) + c^2b \\ c^2f + sc(g-a) - s^2b & s^2a - sc(f+b) + c^2g \end{bmatrix} = \begin{bmatrix} a' & b' \\ 0 & g' \end{bmatrix}$$

Thus,  $c^2 f + sc(g - a) - s^2 b = 0$ . This can be manipulated into

$$\cos(2\phi) \left( \frac{f+b}{2} \right) + \sin(2\phi) \left( \frac{g-a}{2} \right) = \left( \frac{b-f}{2} \right).$$

To solve the above, we can consider it like a linear differential equation— there is a homogeneous part and a particular part. We can write:

$$\begin{aligned} \cos(2\phi) &= K_1(a - g) + K_2(f + b) \\ \sin(2\phi) &= K_1(f + b) + K_2(g - a) \end{aligned}$$

with

$$\begin{aligned} K_2 &= \frac{b-f}{(f+b)^2 + (g-a)^2} \\ K_1 &= \frac{[(g-a)^2 + 4bf]^{1/2}}{[(f+b)^2 + (g-a)^2]}. \end{aligned}$$

$K_2$  is associated with the “particular” portion and scales it to be equal to the term  $\frac{b-f}{2}$ .  $K_1$  normalizes  $\cos^2(2\phi) + \sin^2(2\phi) = 1$ . Note that one has the condition:  $(g-a)^2 + 4bf \geq 0$  which complements the condition facing the transformation converting 2x2 blocks into the  $\sigma - \omega$  form. In other words, if this condition is not met, it is a complex eigenvalue pair.

## A.4 Reducing the Schur Matrix to the Diagonal Form

One can subdivide the problem of zeroing the upper triangular portion of a Schur matrix into 4 cases that arise when one tries to zero the cross-term between roots. All of these cases are just specifications of the general form specified in [27]. When a matrix has been Schur decomposed there will, in the most general form, be an upper triangular partition along with a diagonal of real roots and 2x2 complex pairs. A *normal* matrix will be diagonalized by a Schur decomposition— the Frobenius norm of the strictly upper triangular partition of the matrix is known as a matrix’s *departure from normality* (see [23]). To diagonalize the upper triangular matrix we have to use nonunitary similarity transformations. The most convenient form is

$$R = \begin{bmatrix} \mathbf{I} & P \\ 0 & \mathbf{I} \end{bmatrix}; \quad R^{-1} = \begin{bmatrix} \mathbf{I} & -P \\ 0 & \mathbf{I} \end{bmatrix};$$

Suppose that one were to zero the upper triangular part between two diagonal submatrices with this transformation:

$$\begin{bmatrix} \mathbf{I} & -P \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Lambda_1 & T \\ 0 & \Lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{I} & P \\ 0 & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \Lambda_1 & T - P\Lambda_2 + \Lambda_1 P \\ 0 & \Lambda_2 \end{bmatrix}$$

In general, one has the Lyapunov equation:  $T - P\Lambda_2 + \Lambda_1 P = 0$  to solve. This equation has a solution provided that there are no eigenvalues in  $\Lambda_1$  equal to those in  $\Lambda_2$ . However, in general, the presence of degenerate eigenvalues, whether defective or not, will make determination of this transformation impossible. Not only that, but in numerical computation defectiveness is more of a condition to be approached in a continuous manner rather than an

on-off setting. A measure of this more continuous form of defectiveness is in the condition of the transformation (diagonalizing) matrix. The condition of the overall transformation matrix will be helped by limiting the condition allowed for these individual non-unitary transformations. By declaring a minimum allowed condition, one thus detects, in a sense, which eigenvalues are part of a diagonalizable submatrix and which are not. (Because the inverse is explicitly known, this condition is easy to estimate.)

It may actually be that a pair of distinct (but close) eigenvalues will get declared a degenerate pair, but this will only impose a slight time penalty on the eventual calculations. It can be said that the resolution on the eigenvalues was not sufficient to consider them as distinct.

Fortunately, non-defective degenerate eigenvalues form a sort of normal submatrix so that their cross-terms are already zero as a result of the Schur decomposition—zero being the effective zero for a given machine precision, the  $\infty$  - *norm* of the matrix, and the machine zero given the algorithm for the subdiagonal elements. It is possible to detect this relative zero condition and avoid problems in solving for a transformation matrix—by not having to solve for a transformation matrix at all.

The end result will be the diagonalization of the non-defective eigenvalues, the continuance of an upper triangular block for the defective eigenvalues, and a transformation that will remain well-conditioned. What becomes necessary, however, to keep the problem tractable is to break the cross-term zeroing into 4 cases: 2 real roots; a real root and a complex pair; a complex pair and a real root; and finally, between two complex pairs.

## A.5 Reducing the Upper Triangular Part Between 2 Real Roots

Reducing the upper triangular part between 2 real roots is the first (and easiest) case. One literally has

$$\begin{bmatrix} 1 & -p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 & t_{12} \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} 1 & p \\ 0 & 1 \end{bmatrix},$$

resulting in  $p = \frac{t_{12}}{\lambda_2 - \lambda_1}$ . One does not need to solve for  $p$  if  $t_{12}$  is relative zero for the matrix (thus, it would not matter if  $\lambda_1 = \lambda_2$ ).

## A.6 Upper Triangular Part Between a Real Root and a Complex Pair

This case is elucidated separately mostly as a means of showing notation. The submatrices of interest would be

$$\begin{bmatrix} 1 & -p_{11} & -p_{12} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_{11} & t_{11} & t_{12} \\ 0 & \Lambda_{11} & \Lambda_{12} \\ 0 & \Lambda_{21} & \Lambda_{22} \end{bmatrix} \begin{bmatrix} 1 & p_{11} & p_{12} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Zeroing  $t_{11}$  and  $t_{12}$  results in the following system of equations

$$\begin{bmatrix} \Lambda_{11} - \lambda_{11} & \Lambda_{21} \\ \Lambda_{12} & \Lambda_{22} - \lambda_{11} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \end{bmatrix} = \begin{bmatrix} t_{11} \\ t_{12} \end{bmatrix}$$

which *should* always be solveable. In any case, one could always check for a zero solution here as well.

## A.7 Upper Triangular Part Between a Complex Pair and a Real Root

This case is distinct from the above due to a slight difference in equations and notation

$$\begin{bmatrix} 1 & 0 & -p_{11} \\ 0 & 1 & -p_{21} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Lambda_{11} & \Lambda_{12} & t_{11} \\ \Lambda_{21} & \Lambda_{22} & t_{21} \\ 0 & 0 & \lambda_{11} \end{bmatrix} \begin{bmatrix} 1 & 0 & p_{11} \\ 0 & 1 & p_{21} \\ 0 & 0 & 1 \end{bmatrix}$$

Zeroing  $t_{11}$  and  $t_{21}$  results in the following system of equations

$$\begin{bmatrix} \Lambda_{11} - \lambda_{11} & -\lambda_{12} \\ -\lambda_{21} & \Lambda_{11} - \lambda_{22} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{21} \end{bmatrix} = \begin{bmatrix} t_{11} \\ t_{21} \end{bmatrix}$$

## A.8 Upper Triangular Block Between 2 Complex Pairs

Here one solves for the 2x2 block that zeros the upper triangular portion between a set of complex pairs (not necessarily in  $\sigma - \omega$  form)

$$\begin{bmatrix} 1 & 0 & -p_{11} & -p_{12} \\ 0 & 1 & -p_{21} & -p_{22} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_{11} & \lambda_{12} & t_{11} & t_{12} \\ \lambda_{21} & \lambda_{22} & t_{21} & t_{22} \\ 0 & 0 & \Lambda_{11} & \Lambda_{12} \\ 0 & 0 & \Lambda_{21} & \Lambda_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 & p_{11} & p_{12} \\ 0 & 1 & p_{21} & p_{22} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The accompanying system of equations is

$$\begin{bmatrix} \Lambda_{11} - \lambda_{11} & \Lambda_{21} & -\lambda_{12} & 0 \\ \Lambda_{12} & \Lambda_{22} - \lambda_{11} & 0 & -\lambda_{12} \\ -\lambda_{21} & 0 & \Lambda_{11} - \lambda_{22} & \Lambda_{21} \\ 0 & -\lambda_{21} & \Lambda_{12} & \Lambda_{22} - \lambda_{22} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{21} \\ p_{22} \end{bmatrix} = \begin{bmatrix} t_{11} \\ t_{12} \\ t_{21} \\ t_{22} \end{bmatrix}$$

## A.9 Using These Techniques for Zeroing Blocks

The emphasis here is to make sure that one can separate non-defective eigenvalues (and associated eigenvectors) from a blocks containing defective ones. While physically separating individual eigenvalues (or complex pairs) from each other and from defective degenerate blocks while zeroing the upper triangular part between the two may sound appealing, it is easier from the bookkeeping standpoint to keep the diagonal terms *fixed*, then organize them at the end.

Thus one can proceed through the upper triangular part of the matrix in a sequential order, using some sort of indexing array to encode the solitary eigenvalues, the complex pairs, and those eigenvalues belonging to a block of defective degenerate eigenvalues. At each upper triangular element (or block if this element associates one or more complex pairs), one can attempt to zero the members using the methods previously indicated. If a defective degenerate condition is indicated, the association with the eigenvalues will indicate the appropriate block to assign the eigenvalues.

Those eigenvalues whose associated upper triangular parts have been zeroed can be moved past one another in a matrix by a simple shift transformation. The rest will collect

into blocks when they encounter eigenvalues they cannot shift past. These subblocks will be upper triangular and there will be no need to put them into Jordan form.

$$\begin{array}{ccc}
 \text{Schur Form:} & & \text{Decomposed:} \\
 \left[ \begin{array}{cccccc}
 \lambda_1 & u_{12} & \cdots & \cdots & u_{1,n} \\
 & \ddots & \ddots & & \vdots \\
 & & \lambda_i & \ddots & \vdots \\
 & & & \ddots & u_{n-1,n} \\
 & 0 & & & \lambda_n
 \end{array} \right] & \longrightarrow & \left[ \begin{array}{cccccc}
 \ddots & & 0 & & & 0 \\
 & \Lambda & & & 0 & \\
 0 & & \ddots & & & \\
 & & & \boxed{B_1} & & 0 \\
 & 0 & & & \boxed{B_2} & \\
 0 & & & 0 & & \ddots
 \end{array} \right]
 \end{array}$$

## Appendix B

# Cost and Gradient Computation Using The Hybrid Algorithm

### B.1 Introduction

The current two methods within SANDY represent two extremes in the handling of matrix functions: the faster original “diagonal” method decomposed the argument matrices into eigenvalues and eigenvectors, then applied scalar functions to the diagonal terms; while the nominally more robust method used an exponential series to compute a function of the whole matrix.

While the diagonal method depends on successfully determining an eigenvalue–eigenvector decomposition, the robust form also has a weakness—precisely where the diagonal method is best suited. If one has scales in a matrix that are wildly varying, the robust form is more prone to error, even to the point where a defective-degeneracy does not look so bad. Consider a root matrix

$$\Lambda = \begin{bmatrix} -1.0E-8 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1.0E-8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 18 & 1 & 0 & 0 \\ 0 & 0 & 0 & 18 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.0E+8 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1.0E+8 \end{bmatrix}.$$

This matrix is transformed by the following matrix:

$$T = \begin{bmatrix} 0.2113 & 0.4524 & 0.6538 & 0.7469 & 0.1167 & 0.2260 \\ 0.0824 & 0.8075 & 0.4899 & 0.0378 & 0.6250 & 0.8159 \\ 0.7599 & 0.4832 & 0.7741 & 0.4237 & 0.5510 & 0.2284 \\ 0.0087 & 0.6135 & 0.9626 & 0.2613 & 0.3550 & 0.8553 \\ 0.8096 & 0.2749 & 0.9933 & 0.2403 & 0.4943 & 0.0621 \\ 0.8474 & 0.8807 & 0.8360 & 0.3405 & 0.0365 & 0.7075 \end{bmatrix}.$$

The condition of this matrix is 51. This should not introduce any complications in this comparison of methods. Because fundamentally all calculations depend on taking an exponentiation a good test case can be founded on the comparison of exponentiation calculations. Matrix exponentiation of the component submatrices of  $\Lambda$  and transforming by  $T\Lambda T^{-1}$  gives

the most accurate result,

$$1.0E + 8 \times \begin{bmatrix} 0.4628 & -1.2715 & 1.9936 & 1.0176 & -1.4230 & -0.4304 \\ 0.3284 & -0.3895 & 0.2382 & 0.3651 & -0.1007 & -0.1652 \\ 0.5318 & -1.0083 & 1.2521 & 0.8541 & -0.8324 & -0.3708 \\ 0.6519 & -0.9672 & 0.9181 & 0.8599 & -0.5438 & -0.3810 \\ 0.6717 & -0.9664 & 0.8768 & 0.8650 & -0.5069 & -0.3843 \\ 0.5702 & -0.9626 & 1.0705 & 0.8333 & -0.6824 & -0.3651 \end{bmatrix}.$$

Suppose one transforms  $\Lambda$  and then exponentiates, the difference with the previous calculation would be

$$\begin{bmatrix} -158.9406 & 175.7559 & 8.2702 & -210.1598 & -70.0591 & 84.2343 \\ -22.1443 & 57.9680 & -55.9843 & -60.4853 & 30.5230 & 21.5482 \\ -95.5913 & 147.6670 & -69.7673 & -164.4468 & 11.1770 & 63.0574 \\ -70.9993 & 145.2202 & -113.4331 & -154.9395 & 51.9346 & 57.2788 \\ -58.3577 & 151.9611 & -153.7657 & -156.0765 & 86.3283 & 56.7881 \\ -83.6180 & 141.6097 & -81.9972 & -155.5265 & 24.5134 & 58.7931 \end{bmatrix}.$$

This error is still only on the order of one part in  $10^6$ . An exponentiation arrived at by the diagonal method does not have as much error though the matrix is defective degenerate,

$$T = \begin{bmatrix} -19.0856 & 23.9609 & -13.3939 & -24.0870 & 4.2366 & 10.6407 \\ -6.1430 & 5.6210 & 0.5094 & -6.3906 & -2.4559 & 2.9409 \\ -15.4594 & 17.7026 & -6.9271 & -18.3993 & 0.3217 & 8.2278 \\ -15.4252 & 16.3490 & -3.9463 & -17.4740 & -2.0148 & 7.9009 \\ -15.4366 & 16.2809 & -3.7615 & -17.4420 & -2.1720 & 7.8944 \\ -15.2599 & 17.0281 & -5.8801 & -17.8393 & -0.4249 & 8.0123 \end{bmatrix}.$$

The condition of the eigenvector matrix is  $9.9927E + 4$  due to the defectiveness, thus the calculation is probably not too impaired.

Thus, it is hoped that, in addition to an increase in calculation speed, this algorithm will be made more complete.

## B.2 The $\mathcal{X}$ Integral

The simpler of these integrals is  $\mathcal{X}(t) = \int_0^t e^{A\tau} B e^{C\tau} d\tau$ . The matrices  $A$  and  $C$  can be decomposed along the lines of an eigenvalue—eigenvector decomposition, but only for the non-degenerate eigenvalues. The degenerate eigenvalues are decoupled from the non-degenerate eigenvalues, but are otherwise left in a non-diagonalized matrix. The non-diagonalized part of the matrix will consist of decoupled blocks of degenerate sets of eigenvalues. This decomposition is based on a Schur decomposition with a selective eigenvalue shift and a decoupling algorithm. We have

$$A = V_A \begin{bmatrix} \Lambda_A & 0 \\ 0 & W_A \end{bmatrix} V_A^{-1} \text{ and } C = V_C \begin{bmatrix} \Lambda_C & 0 \\ 0 & W_C \end{bmatrix} V_C^{-1}$$

with  $\Lambda_A$  and  $\Lambda_C$  both being diagonal. For the exponential

$$e^{At} = \exp \left\{ V_A \begin{bmatrix} \Lambda_A & 0 \\ 0 & W_A \end{bmatrix} V_A^{-1} t \right\} = V_A \exp \left\{ \begin{bmatrix} \Lambda_A & 0 \\ 0 & W_A \end{bmatrix} t \right\} V_A^{-1}.$$

This results in the following form for  $\mathcal{X}$

$$\begin{aligned}\mathcal{X}(t) &= \int_0^t e^{A\tau} B e^{C\tau} d\tau \\ &= V_A \left\{ \int_0^t \exp \left\{ \begin{bmatrix} \Lambda_A & 0 \\ 0 & W_A \end{bmatrix} \tau \right\} \mathcal{B} \exp \left\{ \begin{bmatrix} \Lambda_C & 0 \\ 0 & W_C \end{bmatrix} \tau \right\} d\tau \right\} V_C^{-1} \quad (\text{B.1})\end{aligned}$$

where  $\mathcal{B} = V_A^{-1} B V_C$ . The exponentials can be expanded and  $\mathcal{X}$  can be separated into parts involving only the non-defective eigenvalues, the defective degenerate part, and combinations of both. For

$$\mathcal{B} = \begin{bmatrix} \mathcal{B}_{11} & \mathcal{B}_{12} \\ \mathcal{B}_{21} & \mathcal{B}_{22} \end{bmatrix} \quad (\text{B.2})$$

we have

$$\mathcal{X}(t) = V_A \begin{bmatrix} \int_0^t e^{\Lambda_A \tau} \mathcal{B}_{11} e^{\Lambda_C \tau} d\tau & \int_0^t e^{\Lambda_A \tau} \mathcal{B}_{12} e^{W_C \tau} d\tau \\ \int_0^t e^{W_A \tau} \mathcal{B}_{21} e^{\Lambda_C \tau} d\tau & \int_0^t e^{W_A \tau} \mathcal{B}_{22} e^{W_C \tau} d\tau \end{bmatrix} V_C^{-1} \quad (\text{B.3})$$

As a result one can use the algorithm used in [18] for the upper left partition (usually all or most of the matrix), and thus preserve speed. The remaining portions involve various uses of the exponential to represent the integral.

Consider the similar forms for the integrals of the upper right and lower left blocks. These involve a combination of a diagonal portion and an irreducible portion (typically of dimension much smaller than the original matrix). For example, one can rewrite the upper right integral in the following way

$$\mathcal{X}_{12}(t) = \int_0^t e^{\Lambda_A \tau} \mathcal{B}_{12} e^{W_C \tau} d\tau \quad (\text{B.4})$$

with

$$\mathcal{X}_{kw}(t) \equiv \text{the } k^{\text{th}} \text{ row of } \mathcal{X}_{12} \quad (\text{B.5})$$

$$= \int_0^t e^{\lambda_{Ak} \tau} b_{kW} e^{W_C \tau} d\tau \quad (\text{B.6})$$

For when  $\lambda_{Ak}$  is a scalar,  $b_{kW}$  is the  $k^{\text{th}}$  row of  $\mathcal{B}_{12}$ . The number of columns in  $b_{kW}$  corresponds to the number of columns of the degenerate matrix  $W_C$ . Because these matrix integrals can be broken down row by row, in the case of the upper right block, or column by column, in the case of the lower left block, there is a considerable time savings. Computation of the matrix exponential corresponding to the integral  $\int_0^t e^{A\tau} B e^{C\tau} d\tau$  for a general  $A$ ,  $B$ , and  $C$  is on the order of  $(\dim(A) + \dim(C))^3$  floating point operations. Breaking down this integral by rows (or columns) results in the advantage of evaluating several small integrals. This special integral form allows us to write:

$$\mathcal{X}_{kw}(t) = \int_0^t b_{kW} e^{(W_C + \lambda_{Ak} I) \tau} d\tau$$

An algorithm for evaluating the matrix exponential integral will be presented in section B.4.

For  $\lambda_{Ak}$  being a 2x2 complex root pair block, the situation is a bit more complicated. Since

$$\exp \begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix} = e^\sigma \begin{bmatrix} \cos \omega & \sin \omega \\ -\sin \omega & \cos \omega \end{bmatrix},$$

one can expand the integral (B.6):

$$\begin{aligned}\mathcal{X}_{kw}(t)[col1] &= \int_0^t e^{\sigma_{Ak}\tau} \cos(\omega_{Ak}\tau) b_{kW} e^{W_C\tau} d\tau + \int_0^t e^{\sigma_{Ak}\tau} \sin(\omega_{Ak}\tau) b_{k+1W} e^{W_C\tau} d\tau \\ \mathcal{X}_{kw}(t)[col2] &= \int_0^t e^{\sigma_{Ak}\tau} \cos(\omega_{Ak}\tau) b_{k+1W} e^{W_C\tau} d\tau - \int_0^t e^{\sigma_{Ak}\tau} \sin(\omega_{Ak}\tau) b_{kW} e^{W_C\tau} d\tau\end{aligned}$$

Again, with  $e^{\sigma_{Ak}\tau}$ ,  $\cos \omega_{Ak}\tau$ , and  $\sin \omega_{Ak}\tau$  as scalars we can re-arrange the above integrals into forms depending on two basic terms:

$$\int_0^t e^{(W_C + \sigma_{Ak}I)\tau} \cos(\omega_{Ak}\tau) d\tau$$

and

$$\int_0^t e^{(W_C + \sigma_{Ak}I)\tau} \sin(\omega_{Ak}\tau) d\tau$$

An algorithm for computing these terms will be presented in section B.5.

The lower right portion (containing only the degenerate eigenvalue matrices) will have to be evaluated with the pure exponential form of the integral evaluation. The size of the degenerate matrix blocks should not be very large relative to the number of non-degenerate eigenvalues. Not only will this keep the overall computation from slowing down, but the eigenvalues shall nominally be of the same relative magnitude.

### B.3 The $\mathcal{M}$ Integral

This integral yields a somewhat more difficult set of cases of integrals to evaluate, as we shall see.

The integral  $\mathcal{M}(t) = \int_0^t \int_0^v e^{A(v-s)} B e^{Cv} D e^{Es} ds dv$  becomes, when  $A$ ,  $C$ , and  $E$  are partitioned into non-defective and defective parts.

$$\begin{aligned}\mathcal{M}(t) &= T_A \int_0^t \int_0^v \begin{bmatrix} e^{\Lambda_A(v-s)} & 0 \\ 0 & e^{W_A(v-s)} \end{bmatrix} \mathcal{B} \begin{bmatrix} e^{\Lambda_C v} & 0 \\ 0 & e^{W_C v} \end{bmatrix} \\ &\quad \mathcal{D} \begin{bmatrix} e^{\Lambda_E s} & 0 \\ 0 & e^{W_E s} \end{bmatrix} ds dv T_E^{-1} \\ &= \left[ \begin{array}{c|c} \int_0^t \int_0^v e^{\Lambda_A(v-s)} \mathcal{B}_{11} e^{\Lambda_C v} \mathcal{D}_{11} e^{\Lambda_E s} ds dv & \int_0^t \int_0^v e^{\Lambda_A(v-s)} \mathcal{B}_{11} e^{\Lambda_C v} \mathcal{D}_{12} e^{W_E s} ds dv \\ + \int_0^t \int_0^v e^{\Lambda_A(v-s)} \mathcal{B}_{12} e^{W_C v} \mathcal{D}_{21} e^{\Lambda_E s} ds dv & + \int_0^t \int_0^v e^{\Lambda_A(v-s)} \mathcal{B}_{12} e^{W_C v} \mathcal{D}_{22} e^{W_E s} ds dv \\ \hline \int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{21} e^{\Lambda_C v} \mathcal{D}_{11} e^{\Lambda_E s} ds dv & \int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{21} e^{\Lambda_C v} \mathcal{D}_{12} e^{W_E s} ds dv \\ + \int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{22} e^{W_C v} \mathcal{D}_{21} e^{\Lambda_E s} ds dv & + \int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{22} e^{W_C v} \mathcal{D}_{22} e^{W_E s} ds dv \end{array} \right]\end{aligned}$$

where

$$\begin{aligned} T_A^{-1} B T_C &= \mathcal{B} = \begin{bmatrix} \mathcal{B}_{11} & \mathcal{B}_{12} \\ \mathcal{B}_{21} & \mathcal{B}_{22} \end{bmatrix} \\ T_C^{-1} D T_E &= \mathcal{D} = \begin{bmatrix} \mathcal{D}_{11} & \mathcal{D}_{12} \\ \mathcal{D}_{21} & \mathcal{D}_{22} \end{bmatrix} \end{aligned}$$

The first part of the  $\mathcal{M}_{11}$  summand can be evaluated in the standard fast way, so it would be desirable to be able to evaluate the second integral as quickly. One can solve analytically for the integral over  $s$ . Let's assume initially that the eigenvalue is real. We have

$$\begin{aligned} & \int_0^t \int_0^v e^{\lambda_i(v-s)} \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} e^{\lambda_j s} ds dv \\ &= \int_0^t e^{\lambda_i v} \left( \int_0^v e^{(\lambda_j - \lambda_i)s} ds \right) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} dv \\ &= \int_0^t \frac{e^{\lambda_j v} - e^{\lambda_i v}}{\lambda_j - \lambda_i} \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} dv \\ &= \frac{\mathcal{B}_{i12}}{\lambda_j - \lambda_i} \int_0^t \left( e^{(W_C + \lambda_j I)v} - e^{(W_C + \lambda_i I)v} \right) dv \mathcal{D}_{21j} \end{aligned}$$

The end result is just the matrix exponential integral again. For a complex pair, one has to deal with a 2x2 block (looking just at the integrand)

$$\begin{bmatrix} e^{\sigma_i(v-s)} \cos \omega_i(v-s) \mathcal{B}_{i12} + e^{\sigma_i(v-s)} \sin \omega_i(v-s) \mathcal{B}_{i+1,12} \\ e^{\sigma_i(v-s)} \cos \omega_i(v-s) \mathcal{B}_{i+1,12} - e^{\sigma_i(v-s)} \sin \omega_i(v-s) \mathcal{B}_{i12} \end{bmatrix} e^{W_C v}$$

$$\begin{bmatrix} \mathcal{D}_{21j} e^{\sigma_j s} \cos \omega_j s & | & \mathcal{D}_{21j+1} e^{\sigma_i s} \cos \omega_j s \\ -\mathcal{D}_{21j+1} e^{\sigma_j s} \sin \omega_j s & | & \mathcal{D}_{21j} e^{\sigma_i s} \cos \omega_j s \end{bmatrix}$$

This becomes

$$e^{\sigma_i v} e^{(\sigma_j - \sigma_i)s}$$

$$\begin{bmatrix} \begin{aligned} & \left[ \cos \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} \right. \\ & \left. + \sin \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j} \right] \\ & \cdot \cos \omega_j s \\ & - \left[ \cos \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21,j+1} \right. \\ & \left. + \sin \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21,j+1} \right] \\ & \cdot \sin \omega_j s \end{aligned} & | & \begin{aligned} & \left[ \cos \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21,j+1} \right. \\ & \left. + \sin \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21,j+1} \right] \\ & \cdot \cos \omega_j s \\ & + \left[ \cos \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} \right. \\ & \left. + \sin \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j} \right] \\ & \cdot \sin \omega_j s \end{aligned} \\ \hline \begin{aligned} & \left[ \cos \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j} \right. \\ & \left. - \sin \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} \right] \\ & \cdot \cos \omega_j s \\ & - \left[ \cos \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j+1} \right. \\ & \left. - \sin \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j+1} \right] \\ & \cdot \sin \omega_j s \end{aligned} & | & \begin{aligned} & \left[ \cos \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j+1} \right. \\ & \left. - \sin \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j+1} \right] \\ & \cdot \cos \omega_j s \\ & \left[ \cos \omega_i(v-s) \mathcal{B}_{i+1,12} e^{W_C v} \mathcal{D}_{21j} \right. \\ & \left. - \sin \omega_i(v-s) \mathcal{B}_{i12} e^{W_C v} \mathcal{D}_{21j} \right] \\ & \cdot \sin \omega_j s \end{aligned} \end{bmatrix}$$

The integrals over  $s$  will boil down to these terms

$$\begin{aligned} \int_0^v e^{(\sigma_j - \sigma_i)s} \cos(\omega_j - \omega_i)s \, ds & \quad \int_0^v e^{(\sigma_j - \sigma_i)s} \cos(\omega_j + \omega_i)s \, ds \\ \int_0^v e^{(\sigma_j - \sigma_i)s} \sin(\omega_j - \omega_i)s \, ds & \quad \int_0^v e^{(\sigma_j - \sigma_i)s} \sin(\omega_j + \omega_i)s \, ds \end{aligned}$$

Substitution of these back in for our 2x2 matrix will yield a series of integrals of the form

$$\int_0^t e^{(W_C + \sigma' I)v} \cos \omega' v \, dv$$

and

$$\int_0^t e^{(W_C + \sigma' I)v} \sin \omega' v \, dv$$

The amount of algebra to get this matrix into a series of integrals on  $v$  is substantial.

Being able to re-express the integral over  $s$  in analytic form is not only sufficient for simplifying the double integral but *necessary*. Changing the order of integration will introduce numerically unstable terms that can be handled only by doing the whole double integral numerically in the current “robust”  $\mathcal{M}$  calculation. Thus one needs to have only simple eigenvalues involved in the  $s$  integral. This happens only on the integrals in the  $\mathcal{M}_{11}$  partition. The rest of the integrals will have to use the current robust form in some manner.

One could be given the impression that not much work and time is being saved. However, a close examination of what happens when one does these other partitions with the robust algorithm one will get a flavor for the amount of time saved.

Consider the sum for  $\mathcal{M}_{22}$ . With the following

$$\int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{21} e^{\Lambda_C v} \mathcal{D}_{12} e^{W_E s} \, ds \, dv = \sum_i \int_0^t \int_0^v e^{W_A(v-s)} \mathcal{B}_{21i} e^{\lambda_{Cii} v} \mathcal{D}_{i12} e^{W_E s} \, ds \, dv$$

Considering that the overall cost of a direct matrix exponentiation is of  $\mathcal{O}(n^3)$  or more, there is a net win. The fact that  $\lambda_{Cii}$  may represent a complex pair in the form  $\begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix}$  is of no consequence as the summands will be evaluated by a direct matrix exponentiation.

## B.4 Padé Series for Matrix Exponential Integral

In the hybrid formulation, one of the terms boils down to finding the integral of the exponential of a matrix  $\int_0^t e^{A\tau} \, d\tau$ . It was suggested [23] to compute this integral by taking the following exponential:

$$\exp \left\{ \begin{bmatrix} 0 & \mathbf{I} \\ 0 & A \end{bmatrix} t \right\} = \begin{bmatrix} \mathbf{I} & \int_0^t e^{A\tau} \, d\tau \\ 0 & e^{At} \end{bmatrix}$$

This technique was disputed in [29], where it was determined that a direct Taylor Series for this integral was considered a better method. The basis for this was that the larger overall matrix contributed not only to a lengthy computation time, but also a loss of accuracy. A third alternative is presented here in the use of a Padé Series for computing the integral.

One could call this series the generating function for the integral, but for the fact that it is simpler to assume that one is integrating over a unit interval. Since the matrix contains all the scaling information in a typical function call, it is a safe assumption. We know that:

$$\int_0^1 e^{A\tau} \, d\tau = A^{-1} e^{At} \Big|_0^1 = A^{-1} [e^A - \mathbf{I}]$$

and

$$e^{At} \sim P_M^N(z) = \frac{\sum_{i=0}^N N_i (At)^i}{\sum_{i=0}^M D_i (At)^i}$$

so

$$\int_0^t e^{A\tau} d\tau \sim A^{-1} \frac{\sum_{i=0}^N (N_i - D_i) (At)^i}{\sum_{i=0}^M D_i (At)^i}$$

The most commonly used case of the Padé Series for the exponential is a diagonal sequence (order of the numerator is equal to that of the denominator). For this, a nice simplification in the above equation occurs since, for  $i$  even,  $N_i = D_i$ , and for  $i$  odd,  $N_i = -D_i$ . Thus, we can write the Padé Series for the integral generating function as:

$$\int_0^t e^{A\tau} d\tau \sim t P_M^N(z) = 2t \frac{\sum_{i=1, i=i+2}^N N_i (At)^{(i-1)}}{\sum_{i=0}^N D_i (At)^i}$$

with

$$\begin{aligned} N_i &= \frac{(2N-i)! N!}{(2N)! i! (N-i)!} \\ D_i &= (-1)^i \frac{(2N-i)! N!}{(2N)! i! (N-i)!} \end{aligned}$$

With  $t$  being 1, we have the integral being equal to this particular Padé Series.

Nominally one has to scale the input matrix so that its  $\infty$ -norm is less than  $1/2$ . The rescaling of the result is somewhat different than that of the matrix exponential. The matrix itself doubles in scale, yet the integration interval does not change

$$\begin{aligned} \int_0^1 e^{2At} dt &= \frac{1}{2} \int_0^2 e^{A\tau} d\tau \\ &= \frac{1}{2} \left[ \int_0^1 e^{A\tau} d\tau + \int_1^2 e^{A\tau} d\tau \right] \\ &= \frac{1}{2} \int_0^1 e^{A\tau} d\tau [I + e^A] \end{aligned}$$

The term  $e^A$  is easy to generate from the integral at each step, i.e., multiply  $\int_0^1 e^{A\tau} d\tau$  by the matrix  $A$  and add  $I$ .

## B.5 Padé Series for Integral of Matrix Exponential and Sinusoid

Another special form to handle from the hybrid formulation is

$$\int_0^t e^{A\tau} \sin \omega \tau d\tau \text{ or } \int_0^t e^{A\tau} \cos \omega \tau d\tau.$$

Using the relationship

$$\exp \begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix} = e^\sigma \begin{bmatrix} \cos \omega & \sin \omega \\ -\sin \omega & \cos \omega \end{bmatrix}.$$

Thus we can write

$$\begin{aligned} \exp \begin{bmatrix} W + \sigma I & \omega I \\ -\omega I & W + \sigma I \end{bmatrix} &= \begin{bmatrix} e^{(W+\sigma I)} & 0 \\ 0 & e^{(W+\sigma I)} \end{bmatrix} \begin{bmatrix} I \cos \omega & I \sin \omega \\ -I \sin \omega & I \cos \omega \end{bmatrix} \\ &= \begin{bmatrix} e^{(W+\sigma I)} \cos \omega & e^{(W+\sigma I)} \sin \omega \\ -e^{(W+\sigma I)} \sin \omega & e^{(W+\sigma I)} \cos \omega \end{bmatrix} \end{aligned}$$

It would seem that we are doing twice as much work as we need be. Since the augmented  $2n \times 2n$  matrix has several symmetries, one can do specialized sets of multiplications to exploit these symmetries and avoid use of more matrix storage than necessary. In the series calculation, one will need to take squares, etc of the argument matrix

$$\begin{bmatrix} W + \sigma I & \omega I \\ -\omega I & W + \sigma I \end{bmatrix}^2 = \begin{bmatrix} (W + \sigma I)^2 - \omega^2 I & \omega \sigma I \\ -\omega \sigma I & (W + \sigma I)^2 - \omega^2 I \end{bmatrix}$$

One will have only 2 distinct submatrices within this matrix. Thus one need not do explicit multiplies.

## B.6 Scaling

Because all matrix blocks will have their eigenvalues on the diagonal and no sub-diagonal elements, a single scaling parameter characterizes the magnitude of the resulting exponential of the block. For example, in the block

$$\begin{bmatrix} \lambda_1 & u_{12} & \cdots & \cdots & u_{1,n} \\ & \ddots & \ddots & & \vdots \\ & & \lambda_i & \ddots & \vdots \\ & 0 & & \ddots & u_{n-1,n} \\ & & & & \lambda_n \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & & \\ & \ddots & & 0 & \\ & & \lambda_i & & \\ & 0 & & \ddots & \\ & & & & \lambda_n \end{bmatrix} + \begin{bmatrix} 0 & u_{12} & \cdots & \cdots & u_{1,n} \\ & \ddots & \ddots & & \vdots \\ & & 0 & \ddots & \vdots \\ & 0 & & \ddots & u_{n-1,n} \\ & & & & 0 \end{bmatrix}$$

where  $\lambda_1$  through  $\lambda_n$  are the same value. Because

$$e^{Wt} = e^{(\lambda I + W')t} = e^{\lambda t} e^{W't}$$

one can start with an integral like

$$\int_0^t e^{W_A \tau} B e^{W_C \tau} d\tau$$

and “balance” the two matrices into having the same diagonal value. Note that the smaller modulus diagonal terms will probably reduce the number of scaling steps and/or series length in the matrix exponential calculation, thus saving some time.

For the calculation for  $\mathcal{M}$ , the balancing of the matrix blocks is a more complicated job, due to the existence of two separate integrals involving separate matrices

$$\int_0^t \int_0^t e^{A(v-s)} B e^{Cv} D e^{Es} ds dv.$$

However, because the procedure is simple, the attempt is worth it no matter how little results one gets.



## Appendix C

# An Algorithm for SCB

The following sections show different steps in the computation of the SCB transformation discussed in Chapter 5. It is hoped that by proceeding from the simple cases and ending with the more general ones, a better understanding of the algorithm can be achieved.

### C.1 S.C.B. for Non-Strictly Proper SISO Systems

In a SISO system, the presence of a direct feedthrough term essentially eliminates most of the work needed in computing the special coordinate basis. In this case, for the system

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (\text{C.1})$$

where  $u$  and  $y$  are scalar. With  $D \neq 0$ , all the states are basically “internal” states—the input can go to the output directly without necessarily passing through an integrator. One can rewrite the state equations by eliminating  $u$  in terms of  $x$  and  $y$  as follows,

$$\dot{x} = (A - BD^{-1}C)x + BD^{-1}y$$

Clearly there is no longer a direct input term from  $u$  to any of the system states once the substitution has been made. Note that in this case, the system invariant zeros are simply the eigenvalues of the matrix  $(A - BD^{-1}C)$ .

### C.2 S.C.B. for Strictly Proper SISO Systems

Conceptually, this is the next easiest case. With no direct feedthrough term, the task is to determine an output state with a direct feedthrough term through differentiation. Because there is only one input and one output, only “output” states can have a direct input term. Furthermore, the output states or its derivatives will always have a direct input term, indicating that a SISO system is always left and right-invertible. We begin with the system

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (\text{C.2})$$

where  $u$  and  $y$  are scalar. Suppose that one transforms the states using an orthogonal transformation  $V_c$  so that one of the states is the output,

$$C_0 = [c \ 0 \ \cdots \ 0] = CV_C; \quad A_0 = V_C^T A V_C = \begin{bmatrix} A_{0yy} & A_{0yx} \\ A_{0xy} & A_{0xx} \end{bmatrix}$$

and

$$B_0 = V_C^T B = \begin{bmatrix} b_{0y_0} \\ b_{0x_1} \\ b_{0x_2} \\ \vdots \\ b_{0x_{(n-1)}} \end{bmatrix} = \begin{bmatrix} B_{0y} \\ B_{0x} \end{bmatrix}.$$

resulting in the state equations

$$\begin{bmatrix} \dot{y}_0 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} A_{0yy} & A_{0yx} \\ A_{0xy} & A_{0xx} \end{bmatrix} \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} + \begin{bmatrix} B_{0y} \\ B_{0x} \end{bmatrix} u$$

In essence, we have taken one derivative of the output and this also implies that the system has at least one infinite zero.

If the term  $B_{0y}$  (a scalar in a SISO system) is zero, then we continue to generate another output state by differentiating  $\dot{y}_0$ . Namely,

$$\ddot{y}_0 = A_{0yy}\dot{y}_0 + A_{0yx}\dot{x}_0$$

Making the appropriate substitution for  $\dot{x}_0$ , we obtain

$$\ddot{y}_0 = A_{0yy}\dot{y}_0 + A_{0yx} [A_{0xx}x_0 + A_{0xy}y_0 + B_{0x}u]$$

Because  $\dot{y}_0$  is now a state variable, we can write the state equations for  $y_0, \dot{y}_0$  as

$$\dot{y}_1 = \begin{bmatrix} 0 & 1 \\ A_{0yx}A_{0xy} & A_{0yy} \end{bmatrix} y_1 + \begin{bmatrix} 0 \\ A_{0yx}A_{0xx} \end{bmatrix} x_0 + \begin{bmatrix} 0 \\ A_{0yx}B_{0x} \end{bmatrix} u$$

where

$$y_1 = \begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ \dot{y}_0 \end{bmatrix}$$

Note that in SISO systems,  $\bar{y}_1$  always contains the term with the highest derivative, and  $\hat{y}_1$  covers the rest of the output states. In addition, one of the internal states  $x_0$  can be rewritten in terms of  $y_0, \dot{y}_0$  and the remaining states in  $x_0$  using the equation

$$\dot{y}_0 = A_{0yy}y_0 + A_{0yx}x_0 \quad (C.3)$$

Identifying which state among  $x_0$  to be replaced by  $y_0$  depends upon the structure of  $A_{0yx}$ —it is prudent to choose the state corresponding to element in  $A_{0yx}$  having the largest magnitude. Suppose that the  $j^{th}$  element of the state  $x_0$  is chosen. Let's denote this state by  $\bar{x}_0$  and the rest of the internal states in  $x_0$  by  $\hat{x}_0$  with

$$\hat{x}_0 = \begin{bmatrix} x_{01} \\ x_{02} \\ \vdots \\ x_{0(j-1)} \\ x_{0(j+1)} \\ \vdots \\ x_{0(n-1)} \end{bmatrix}$$

Partitioning the state equation for  $x_0$  into separate equations for  $\bar{x}_0$  and  $\hat{x}_0$ , we have

$$\dot{\bar{x}}_0 = A_{0\bar{x}\bar{x}}\bar{x}_0 + A_{0\bar{x}\hat{x}}\hat{x}_0 + A_{0\bar{x}y}y_0 + B_{0\bar{x}}u \quad (C.4)$$

$$\dot{\hat{x}}_0 = A_{0\hat{x}\bar{x}}\bar{x}_0 + A_{0\hat{x}\hat{x}}\hat{x}_0 + A_{0\hat{x}y}y_0 + B_{0\hat{x}}u \quad (C.5)$$

Then using equation (C.3), we can express  $\bar{x}_0$  in terms of the output states  $\hat{y}_1$  and the remaining states  $x_1$  as follows,

$$\dot{y}_0 - A_{0yy}y_0 - A_{0y\hat{x}}\hat{x}_0 = A_{0y\bar{x}}\bar{x}_0,$$

or

$$\bar{x}_0 = A_{0y\bar{x}}^{-1} ([-A_{0yy} \ I] y_1 - A_{0y\hat{x}}x_1)$$

where we define

$$\begin{cases} y_1 &= \begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ \dot{y}_0 \end{bmatrix} \\ x_1 &= \hat{x}_0 \end{cases} \quad (C.6)$$

So far we have replaced two of the system states by output states. The next part of the algorithm can be done recursively.

First, we can re-write the system model in terms of the output states  $y_1$  and the remaining states  $x_1$ . The resulting state equations are

$$\begin{cases} \dot{y}_1 &= A_{1yy}y_1 + A_{1yx}x_1 + B_{1y}u \\ \dot{x}_1 &= A_{1xy}y_1 + A_{1xx}x_1 + B_{1x}u \end{cases} \quad (C.7)$$

where

$$\begin{aligned} A_{1yy} &= \begin{bmatrix} 0 & 1 \\ A_{0yx}A_{0xy} & A_{0yy} \end{bmatrix} + \begin{bmatrix} 0 \\ A_{0yx}A_{0x\bar{x}} \end{bmatrix} A_{0y\bar{x}}^{-1} [-A_{0yy} \ I] \\ A_{1yx} &= \begin{bmatrix} 0 \\ A_{0yx}A_{0x\hat{x}} \end{bmatrix} - \begin{bmatrix} 0 \\ A_{0yx}A_{0x\bar{x}} \end{bmatrix} A_{0y\bar{x}}^{-1} A_{0y\hat{x}} \\ A_{1xy} &= [A_{0\hat{x}y} + A_{0\hat{x}\bar{x}}A_{0y\bar{x}}^{-1} [-A_{0yy} \ I]] \\ A_{1xx} &= A_{0\hat{x}\hat{x}} - A_{0\hat{x}\bar{x}}A_{0y\bar{x}}^{-1} A_{0y\hat{x}} \\ B_{1y} &= \begin{bmatrix} 0 \\ A_{0yx}B_{0x} \end{bmatrix} \\ B_{1x} &= \bar{B}_{0\hat{x}} \end{aligned}$$

Partitioning of  $y_1$  into  $\begin{bmatrix} \bar{y}_1 \\ \hat{y}_1 \end{bmatrix}$  is essential for starting the next iteration. There is no point of taking the derivative of  $\bar{y}_1$  in the formulation of new output states, as they have no direct interaction with the  $x_1$  states. Recall from the first step that we have  $\hat{y}_1 = \dot{y}_0$ . Subsequent iterations would involve repeated differentiation of  $\hat{y}_1$ . Finally, at the  $k^{th}$ -iteration the output state vector  $y_k$  is given by

$$y_k = \begin{bmatrix} \bar{y}_k \\ \hat{y}_k \end{bmatrix}$$

where the output  $\hat{y}_k$  represents the  $k^{th}$  derivative of the output  $y_0$ .

At the  $k^{th}$  iteration, we have

$$\begin{cases} \dot{y}_k &= A_{kyy}y_k + A_{kyx}x_k + B_{ky}u \\ \dot{x}_k &= A_{kxy}y_k + A_{kxx}x_k + B_{kx}u \end{cases} \quad (C.8)$$

where

$$B_{ky} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b_{k\hat{y}} \end{bmatrix} \quad (C.9)$$

$$A_{kyx} = \begin{bmatrix} [0]_{k \times (n-k)} \\ A_{k\hat{y}x} \end{bmatrix} \quad (C.10)$$

$$A_{kyy} = \begin{bmatrix} 0 & & \\ 0 & & \\ \vdots & [I] & \\ 0 & & \\ A_{k\hat{y}\hat{y}} & A_{k\hat{y}y} \end{bmatrix} \quad (C.11)$$

If the term  $B_{ky}$  has some nonzero entry (i.e.,  $b_{k\hat{y}} \neq 0$ ), then the SCB transformation is complete. Otherwise, one must continue to take derivatives of the output state  $\hat{y}_k$  as long as there are nonzero entries in the term  $A_{kyx}$ . Note that both the terms  $B_{ky}$  and  $A_{kyx}$  cannot be zero simultaneously in a SISO system since it is always right-invertible.

When the term  $B_{ky}$  is nonzero, then the  $k^{th}$  derivative of  $y_0$  has a nonzero control input term. The control input  $u$  can be substituted out in terms of the output state, its derivatives and the remaining internal states  $x_k$  as

$$u = 1/b_{k\hat{y}} [\dot{\hat{y}}_k - A_{k\hat{y}\hat{y}}\hat{y}_k - A_{k\hat{y}y}y_k - A_{k\hat{y}x}x_k]$$

This can also be accomplished by performing the following state transformation

$$\begin{bmatrix} y_f \\ x_a \end{bmatrix} = \begin{bmatrix} I & 0 \\ -P & I \end{bmatrix} \begin{bmatrix} y_k \\ x_k \end{bmatrix}$$

where

$$P = B_{kx}(B_{ky}^T B_{ky})^{-1} B_{ky}^T = B_{kx} [0 \ 0 \ \cdots \ 0 \ 1/b_{k\hat{y}}],$$

and

$$\begin{bmatrix} A_{ff} & A_{fa} \\ A_{af} & A_{aa} \end{bmatrix} = \begin{bmatrix} I & 0 \\ -P & I \end{bmatrix} \begin{bmatrix} A_{kyy} & A_{kyx} \\ A_{kxy} & A_{kxx} \end{bmatrix} \begin{bmatrix} I & 0 \\ P & I \end{bmatrix}.$$

After the transformation, the  $u$  input is removed from the state equations associated with the internal states  $x_k$  and we arrive at the final SCB form,

$$\begin{bmatrix} \dot{y}_f \\ \dot{x}_a \end{bmatrix} = \begin{bmatrix} A_{ff} & A_{fa} \\ A_{af} & A_{aa} \end{bmatrix} \begin{bmatrix} y_f \\ x_a \end{bmatrix} + \begin{bmatrix} B_f \\ 0 \end{bmatrix} u$$

The system invariant zeros are simply the eigenvalues of the submatrix  $A_{aa}$ . Clearly, some criterion must be established in testing whether the term  $b_{k\hat{y}}$  at each iteration is zero or near zero. Often, a pre-specified level of tolerance must be given to the SCB algorithm for this singularity test.

### C.3 S.C.B. for Multi-Input and Multi-Output Systems

Let's consider the following system

$$\begin{cases} \dot{x} &= \bar{A}x + \bar{B}\bar{u} \\ \bar{y} &= \bar{C}x + D\bar{u} \end{cases} \quad (\text{C.12})$$

where we assume without loss of generalities that the matrices  $[\bar{B}^T, D^T]^T$  and  $[\bar{C}, D]$  have full rank.

Analogous to the non-strictly proper SISO case, we apply the singular value decomposition to transform the inputs and outputs so that the direct feedthrough term takes on the following form

$$D = U_D \Sigma_D V_D^T = U_D \begin{bmatrix} \sigma_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & \cdots & 0 \\ & & \ddots & & & \\ 0 & \cdots & \cdots & \sigma_k & \cdots & 0 \\ & & & & \ddots & \\ 0 & \cdots & \cdots & \cdots & 0 & \sigma_m \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} V_D^T$$

where  $U_D$  and  $V_D$  are orthogonal transformations and the singular values are given in a decreasing order ( $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$ ). The transformation  $V_D^T$  is applied to the input  $\bar{u}$  while the transformation  $U_D^T$  is to the output  $\bar{y}$ . Further classification and partitioning of the  $x$  states will still be required, as well as for any remaining input and output terms that are not related to the direct feedthrough term. Note that the singular values can be zero for some  $k \leq m$ . With the above input-output transformations, we define

$$\begin{bmatrix} \tilde{u} \\ u \end{bmatrix} = V_D^T \bar{u}$$

and

$$\begin{bmatrix} \tilde{y} \\ y \end{bmatrix} = U_D^T \bar{y}$$

The output equation becomes

$$\begin{bmatrix} \tilde{y} \\ y \end{bmatrix} = \begin{bmatrix} \tilde{C} \\ C \end{bmatrix} x + \begin{bmatrix} \hat{\Sigma}_D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{u} \\ u \end{bmatrix}$$

Partitioning of  $\{\tilde{u}, u\}$  and  $\{\tilde{y}, y\}$  depends on the rank of  $D$ . The matrix  $\hat{\Sigma}_D$  represents the upper-left nonsingular matrix of  $\Sigma_D$ . And the matrix  $\tilde{C}$  and  $C$  are given by

$$\begin{bmatrix} \tilde{C} \\ C \end{bmatrix} = U_D^T \bar{C}$$

We can express  $\tilde{u}$  in terms of the output  $\tilde{y}$  and the system states  $x$  from the equation  $\tilde{y} = \tilde{C}x + \hat{\Sigma}_D \tilde{u}$  as

$$\tilde{u} = \hat{\Sigma}_D^{-1} [\tilde{y} - \tilde{C}x] \quad (\text{C.13})$$

The system given in equation (C.12) can be re-written as

$$\dot{x} = \bar{A}x + \bar{B}V_D \begin{bmatrix} \tilde{u} \\ u \end{bmatrix}$$

or

$$\dot{x} = \bar{A}x + \bar{B}\tilde{u} + Bu$$

Using equation (C.13), we have

$$\begin{cases} \dot{x} &= Ax + Bu + L\tilde{y} \\ \tilde{y} &= \check{C}x + \hat{\Sigma}_D\tilde{u} \\ y &= Cx \end{cases} \quad (\text{C.14})$$

where

$$\begin{aligned} A &= \bar{A} - \bar{B}\hat{\Sigma}_D^{-1}\check{C} \\ L &= \bar{B}\hat{\Sigma}_D^{-1} \end{aligned}$$

First let's consider the case where  $\tilde{u}$  comprises all of the inputs, hence the input  $u$  is non-existent. Furthermore, if the outputs  $y$  is also non-existent, then the SCB transformation is complete resulting in the following system,

$$\begin{cases} \dot{x} &= Ax + L\tilde{y} \\ \tilde{y} &= \check{C}x + \hat{\Sigma}_D\tilde{u} \end{cases} \quad (\text{C.15})$$

Clearly, the system is right- and left-invertible with no infinite zeros. Furthermore, the system invariant zeros are simply the eigenvalues of  $A$ .

Let's examine next the simple case where  $\tilde{y}$  comprises of all the outputs, then the remaining inputs  $u$  must be associated with the internal states  $x$  in the equation

$$\begin{cases} \dot{x} &= Ax + Bu + L\tilde{y} \\ \tilde{y} &= \check{C}x + \hat{\Sigma}_D\tilde{u} \end{cases} \quad (\text{C.16})$$

We can further distinguish the system states  $x$  into two subspaces identified with states  $x_a$  and  $x_c$  corresponding respectively to the uncontrollable and controllable subspaces of the inputs  $u$ , as discussed in Section 5.3. The resulting system is in the following SCB form

$$\begin{bmatrix} \dot{x}_a \\ \dot{x}_c \end{bmatrix} = \begin{bmatrix} A_{aa} & 0 \\ A_{ca} & A_{cc} \end{bmatrix} \begin{bmatrix} x_a \\ x_c \end{bmatrix} + \begin{bmatrix} 0 \\ B_c \end{bmatrix} u + \begin{bmatrix} L_a \\ L_c \end{bmatrix} \tilde{y}$$

where the pair  $(A_{cc}, B_c)$  is controllable. Clearly, the system is right-invertible (but not left-invertible due to the presence of  $u$ ) and has no infinite zeros. In this case, the system invariant zeros are simply the eigenvalues of  $A_{aa}$ .

If on the other hand there are outputs  $y$  in equation (C.14), then they need to be converted into states in a manner similar to that shown in Section C.2 for SISO systems. The procedure will be described in more details in Section C.4.

In the next section, we examine the general case where both  $u$  and  $y$  exist in the equation (C.14).

## C.4 SCB Transformations for Decomposing the Remaining States

Let's consider the system given in equation (C.14) where the direct input  $\tilde{u}$  has already been eliminated through the output  $\tilde{y}$ . We now have the following system,

$$\begin{cases} \dot{x} = Ax + Bu + L\tilde{y} \\ y = Cx \end{cases} \quad (\text{C.17})$$

The task is to resolve the system into the 4 types of SCB states  $y_f$ ,  $y_b$ ,  $x_a$ , and  $x_c$ . The first step follows conceptually the procedure in Section C.2 for SISO systems. We begin by performing the singular value decomposition on the matrix  $C$ ,

$$\begin{aligned} C &= U_C \Sigma V_C^T \\ &= U_C [\hat{\Sigma}_C \ 0] V_C^T \\ &= [\bar{C}_0 \ 0] V_C^T \end{aligned} \quad (\text{C.18})$$

where  $U_C$  and  $V_C$  are orthogonal matrices and  $\hat{\Sigma}_C$  is a nonsingular matrix of dimension  $p - m_0$  which is actually equal to the rank of  $C$ .

At the first iteration  $k = 0$ , we transform the states  $x$  such that some of the states are outputs with the following state transformation

$$\begin{bmatrix} y_0 \\ x_0 \end{bmatrix} = V_C^T x \quad (\text{C.19})$$

Letting  $u_0 = u$ , the state model in equation (C.17) becomes

$$\begin{cases} \begin{bmatrix} \dot{y}_0 \\ \dot{x}_0 \end{bmatrix} = V_C^T A V_C \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} + V_C^T B u_0 + V_C^T L \tilde{y} \\ y = C V_C \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} \end{cases} \quad (\text{C.20})$$

or

$$\begin{cases} \begin{bmatrix} \dot{y}_0 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} A_{0yy} & A_{0yx} \\ A_{0xy} & A_{0xx} \end{bmatrix} \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} + \begin{bmatrix} B_{0y} \\ B_{0x} \end{bmatrix} u_0 + \begin{bmatrix} L_{0y} \\ L_{0x} \end{bmatrix} \tilde{y} \\ y = [\bar{C}_0 \ 0] \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} \end{cases} \quad (\text{C.21})$$

Next, we identify the output states that have direct interaction with  $u_0$  by examining the term  $B_{0y}$ . Again using the singular value decomposition, we have

$$\begin{aligned} B_{0y} &= U_{B_{0y}} \Sigma_{B_{0y}} V_{B_{0y}}^T \\ &= U_{B_{0y}} \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 \\ 0 & 0 \end{bmatrix} V_{B_{0y}}^T \end{aligned} \quad (\text{C.22})$$

We then apply the following transformations to the inputs  $u$  and the outputs  $y_0$ ,

$$U_{B_{0y}}^T y_0 = \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \end{bmatrix} \quad (\text{C.23})$$

and

$$V_{B_{0y}}^T u_0 = \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \end{bmatrix} \quad (\text{C.24})$$

Clearly,

$$\begin{bmatrix} \dot{\hat{y}}_0 \\ \dot{\tilde{y}}_0 \end{bmatrix} = U_{B_{0y}}^T A_{0yy} U_{B_{0y}} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \end{bmatrix} + U_{B_{0y}}^T A_{0yx} x_0 + U_{B_{0y}}^T B_{0y} V_{B_{0y}} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \end{bmatrix} + U_{B_{0y}}^T L_{0y} \tilde{y} \quad (C.25)$$

or

$$\begin{bmatrix} \dot{\hat{y}}_0 \\ \dot{\tilde{y}}_0 \end{bmatrix} = \begin{bmatrix} A_{0y\hat{y}} & A_{0y\tilde{y}} \\ A_{0\tilde{y}\hat{y}} & A_{0\tilde{y}\tilde{y}} \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \end{bmatrix} + \begin{bmatrix} A_{0y\hat{x}} \\ A_{0\tilde{y}\hat{x}} \end{bmatrix} x_0 + \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \end{bmatrix} + \begin{bmatrix} L_{0\hat{y}} \\ L_{0\tilde{y}} \end{bmatrix} \tilde{y} \quad (C.26)$$

and

$$\dot{x}_0 = A_{0xy} U_{B_{0y}} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \end{bmatrix} + A_{0xx} x_0 + B_{0x} V_{B_{0y}} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \end{bmatrix} + L_{0x} \tilde{y} \quad (C.27)$$

or

$$\dot{x}_0 = \begin{bmatrix} A_{0x\hat{y}} & A_{0x\tilde{y}} \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \end{bmatrix} + A_{0xx} x_0 + \begin{bmatrix} B_{0x} \hat{V}_{B_{0y}} & B_{0x} \tilde{V}_{B_{0y}} \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \end{bmatrix} + L_{0x} \tilde{y} \quad (C.28)$$

In the state equation for  $x_0$ , we can eliminate the dependency on  $\hat{u}_0$  using the following equation

$$\hat{u}_0 = \hat{\Sigma}_{B_{0y}}^{-1} \left[ \dot{\hat{y}}_0 - A_{0y\hat{y}} \hat{y}_0 - A_{0y\tilde{y}} \tilde{y}_0 - A_{0y\hat{x}} x_0 - L_{0\hat{y}} \tilde{y} \right]$$

This procedure can also be accomplished by invoking the following state transformation

$$\begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ \tilde{x}_0 \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -P & 0 & I \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ x_0 \end{bmatrix}$$

where

$$P = B_{0x} \hat{V}_{B_{0y}} \hat{\Sigma}_{B_{0y}}^{-1}$$

Let's define the state matrices in the transformed system as

$$\begin{bmatrix} \bar{A}_{0y\hat{y}} & \bar{A}_{0y\tilde{y}} & \bar{A}_{0y\hat{x}} \\ \bar{A}_{0\tilde{y}\hat{y}} & \bar{A}_{0\tilde{y}\tilde{y}} & \bar{A}_{0\tilde{y}\hat{x}} \\ \bar{A}_{0x\hat{y}} & \bar{A}_{0x\tilde{y}} & \bar{A}_{0x\hat{x}} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -P & 0 & I \end{bmatrix} \begin{bmatrix} A_{0y\hat{y}} & A_{0y\tilde{y}} & A_{0y\hat{x}} \\ A_{0\tilde{y}\hat{y}} & A_{0\tilde{y}\tilde{y}} & A_{0\tilde{y}\hat{x}} \\ A_{0x\hat{y}} & A_{0x\tilde{y}} & A_{0x\hat{x}} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ P & 0 & I \end{bmatrix}$$

$$\begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0x} & L_{0x} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -P & 0 & I \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ B_{0x} \hat{V}_{B_{0y}} & B_{0x} \tilde{V}_{B_{0y}} & L_{0x} \end{bmatrix}$$

and

$$\begin{bmatrix} \hat{C}_0 & \tilde{C}_0 & 0 \end{bmatrix} = \begin{bmatrix} \hat{C}_0 & \tilde{C}_0 & 0 \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ P & 0 & I \end{bmatrix}$$

where  $[\hat{C}_0, \tilde{C}_0] = \bar{C}_0 U_{B_{0y}}$ .

The resulting system is given by

$$\begin{cases} \begin{bmatrix} \dot{\hat{y}}_0 \\ \dot{\tilde{y}}_0 \\ \dot{\tilde{x}}_0 \end{bmatrix} = \begin{bmatrix} \bar{A}_{0y\hat{y}} & \bar{A}_{0y\tilde{y}} & \bar{A}_{0y\hat{x}} \\ \bar{A}_{0\tilde{y}\hat{y}} & \bar{A}_{0\tilde{y}\tilde{y}} & \bar{A}_{0\tilde{y}\hat{x}} \\ \bar{A}_{0x\hat{y}} & \bar{A}_{0x\tilde{y}} & \bar{A}_{0x\hat{x}} \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ \tilde{x}_0 \end{bmatrix} + \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0x} & L_{0x} \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \\ \tilde{y} \end{bmatrix} \\ y = \begin{bmatrix} \hat{C}_0 & \tilde{C}_0 & 0 \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ \tilde{x}_0 \end{bmatrix} \end{cases} \quad (C.29)$$

Next, we identify in the state equation for  $\tilde{\bar{y}}_0$  those outputs that have direct influence from  $\bar{x}_0$ . The output vector  $\tilde{\bar{y}}_0$  is then partitioned accordingly into  $\tilde{y}_0$  and  $\tilde{\bar{y}}_0$ . The partition is done based on the rank of  $\bar{A}_{0\tilde{y}\bar{x}}$ . Similar to previous matrix decompositions, we perform the singular value decomposition on the matrix  $\bar{A}_{0\tilde{y}\bar{x}}$ ,

$$\begin{aligned}\bar{A}_{0\tilde{y}\bar{x}} &= U_{0\bar{A}\tilde{y}\bar{x}} \Sigma_{0\bar{A}\tilde{y}\bar{x}} V_{0\bar{A}\tilde{y}\bar{x}}^T \\ &= \begin{bmatrix} \hat{\Sigma}_{\bar{A}_{0\tilde{y}\bar{x}}} & 0 \\ 0 & 0 \end{bmatrix}\end{aligned}\quad (\text{C.30})$$

where  $U_{0\bar{A}\tilde{y}\bar{x}}$  and  $V_{0\bar{A}\tilde{y}\bar{x}}$  are orthogonal matrices. Let's define the following state transformation

$$\begin{bmatrix} \hat{\bar{x}}_0 \\ \tilde{\bar{x}}_0 \end{bmatrix} = V_{0\bar{A}\tilde{y}\bar{x}}^T \bar{x}_0 \quad (\text{C.31})$$

and the following output transformation

$$\begin{bmatrix} \tilde{y}_0 \\ \tilde{\bar{y}}_0 \end{bmatrix} = U_{0\bar{A}\tilde{y}\bar{x}}^T \tilde{\bar{y}}_0 \quad (\text{C.32})$$

The output state equation for  $\tilde{\bar{y}}$  is now partitioned into the following output state equations

$$\begin{cases} \dot{\tilde{y}}_0 &= \bar{A}_{0\tilde{y}\tilde{y}} \tilde{y}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{y}}} \tilde{\bar{y}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{x}}} \hat{\bar{x}}_0 + L_{0\tilde{y}} \tilde{y} \\ \dot{\tilde{\bar{y}}}_0 &= \bar{A}_{0\tilde{\bar{y}}\tilde{y}} \tilde{y}_0 + \bar{A}_{0\tilde{\bar{y}}\tilde{\bar{y}}} \tilde{\bar{y}}_0 + \bar{A}_{0\tilde{\bar{y}}\tilde{\bar{x}}} \hat{\bar{x}}_0 + L_{0\tilde{\bar{y}}} \tilde{\bar{y}} \end{cases} \quad (\text{C.33})$$

where  $\hat{\Sigma}_{\bar{A}_{0\tilde{y}\bar{x}}}$  is a square and invertible submatrix of  $\Sigma_{0\bar{A}\tilde{y}\bar{x}}$ .

By taking derivatives of the  $\tilde{y}_0$  states, the algorithm converts system states  $\bar{x}_0$  into output states. This technique will be repeated on subsequent steps until the overlap indicated by the matrix  $\bar{A}_{k\tilde{y}\bar{x}}$  is zero at some  $k^{th}$  iteration.

Note that if there is no direct feedthrough term  $L_{0\tilde{y}}$  in the  $\dot{\tilde{y}}_0$  state equation, then one can simply take its derivative as follows,

$$\ddot{\tilde{y}}_0 = \bar{A}_{0\tilde{y}\tilde{y}} \dot{\tilde{y}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{y}}} \dot{\tilde{\bar{y}}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{x}}} \dot{\hat{\bar{x}}}_0 + \hat{\Sigma}_{0\bar{A}\tilde{y}\bar{x}} \dot{\hat{\bar{x}}}_0 \quad (\text{C.34})$$

thereby creating  $\dot{\tilde{y}}_0$  as a new set of output states replacing  $\hat{\bar{x}}_0$ . Of course, we need to substitute  $\dot{\tilde{y}}_0$ ,  $\dot{\tilde{\bar{y}}}_0$ , and  $\dot{\hat{\bar{x}}}_0$  from equations (C.29) and (C.33) in the above equation. The term  $\dot{\tilde{y}}_0$  is not substituted out because it is no longer part of the state equations—this equation is used to replace  $\hat{\bar{x}}_0$  states with  $\dot{\tilde{y}}_0$  states.

However, if the direct feedthrough term  $L_{0\tilde{y}}$  is not zero, then the term  $L_{0\tilde{y}} \dot{\tilde{y}}$  would be present in the equation for  $\ddot{\tilde{y}}_0$  in equation (C.34). It is impossible to create  $\dot{\tilde{y}}$  since this term would include a time derivative of the control term  $\tilde{u}$ . However, instead of creating the states  $\dot{\tilde{y}}_0$ , one would consider a new output

$$\begin{aligned}\check{\tilde{y}}_0 &= \dot{\tilde{y}}_0 - L_{0\tilde{y}} \tilde{y} \\ &= \bar{A}_{0\tilde{y}\tilde{y}} \tilde{y}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{y}}} \tilde{\bar{y}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{x}}} \hat{\bar{x}}_0 + \hat{\Sigma}_{0\bar{A}\tilde{y}\bar{x}} \hat{\bar{x}}_0\end{aligned}\quad (\text{C.35})$$

Differentiating the above equation we have

$$\dot{\check{\tilde{y}}}_0 = \bar{A}_{0\tilde{y}\tilde{y}} \dot{\tilde{y}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{y}}} \dot{\tilde{\bar{y}}}_0 + \bar{A}_{0\tilde{y}\tilde{\bar{x}}} \dot{\hat{\bar{x}}}_0 + \hat{\Sigma}_{0\bar{A}\tilde{y}\bar{x}} \dot{\hat{\bar{x}}}_0 \quad (\text{C.36})$$

Note that we have a new output state equation from the above definition of  $\check{\tilde{y}}_0$

$$\dot{\check{\tilde{y}}}_0 = \check{\tilde{y}}_0 + L_{0\tilde{y}} \tilde{y} \quad (\text{C.37})$$

and equation (C.36) becomes

$$\dot{\tilde{y}}_0 = \bar{A}_{0\tilde{y}\tilde{y}}\dot{\tilde{y}}_0 + \bar{A}_{0\tilde{y}\tilde{y}}[\tilde{y}_0 + L_{0\tilde{y}}\tilde{y}] + \bar{A}_{0\tilde{y}\tilde{y}}\dot{\tilde{y}}_0 + \hat{\Sigma}_{0\tilde{A}\tilde{y}\tilde{x}}\dot{\tilde{x}}_0 \quad (C.38)$$

Substituting the appropriate equations for  $\dot{\tilde{y}}_0$ ,  $\dot{\tilde{y}}_0$  and  $\dot{\tilde{x}}_0$  into equation (C.38), we obtain

$$\begin{aligned} \dot{\tilde{y}}_0 &= \bar{A}_{0\tilde{y}\tilde{y}}[\tilde{y}_0 + L_{0\tilde{y}}\tilde{y}] \\ &+ \bar{A}_{0\tilde{y}\tilde{y}}[\bar{A}_{0\tilde{y}\tilde{y}}\hat{y}_0 + \bar{A}_{0\tilde{y}\tilde{y}}\bar{y}_0 + \bar{A}_{0\tilde{y}\tilde{y}}\tilde{y}_0 + \bar{A}_{0\tilde{y}\tilde{x}}\hat{x}_0 + \bar{A}_{0\tilde{y}\tilde{x}}\tilde{x}_0 + \hat{\Sigma}_{B_{0y}}\hat{u}_0 + L_{0\tilde{y}}\tilde{y}] \\ &+ \bar{A}_{0\tilde{y}\tilde{y}}[\bar{A}_{0\tilde{y}\tilde{y}}\hat{y}_0 + \bar{A}_{0\tilde{y}\tilde{y}}\bar{y}_0 + \bar{A}_{0\tilde{y}\tilde{y}}\tilde{y}_0 + L_{0\tilde{y}}\tilde{y}] \\ &+ \hat{\Sigma}_{\bar{A}_{0\tilde{y}\tilde{x}}}[\bar{A}_{0\tilde{x}\tilde{y}}\hat{y}_0 + \bar{A}_{0\tilde{x}\tilde{y}}\bar{y}_0 + \bar{A}_{0\tilde{x}\tilde{y}}\tilde{y}_0 + \bar{A}_{0\tilde{x}\tilde{x}}\hat{x}_0 + \bar{A}_{0\tilde{x}\tilde{x}}\tilde{x}_0 + \tilde{B}_{0\tilde{x}}\tilde{u}_0 + L_{0\tilde{x}}\tilde{y}] \end{aligned}$$

After simplification, we have

$$\dot{\tilde{y}}_0 = A_{0\tilde{y}\tilde{y}}\hat{y}_0 + A_{0\tilde{y}\tilde{y}}\bar{y}_0 + A_{0\tilde{y}\tilde{y}}\tilde{y}_0 + A_{0\tilde{y}\tilde{y}}\tilde{y}_0 + A_{0\tilde{y}\tilde{x}}\hat{x}_0 + A_{0\tilde{y}\tilde{x}}\tilde{x}_0 + \hat{B}_{0\tilde{y}}\hat{u}_0 + \tilde{B}_{0\tilde{y}}\tilde{u}_0 + L_{0\tilde{y}}\tilde{y} \quad (C.39)$$

Using equation (C.35), we can solve for  $\hat{x}_0$  in terms of  $\tilde{y}_0$ ,  $\hat{y}_0$ ,  $\bar{y}_0$ ,  $\tilde{y}_0$  and  $\tilde{y}$ . Namely,

$$\hat{x}_0 = \hat{\Sigma}_{0\tilde{A}\tilde{y}\tilde{x}}^{-1}[\tilde{y}_0 + L_{0\tilde{y}}\tilde{y} - \bar{A}_{0\tilde{y}\tilde{y}}\hat{y}_0 - \bar{A}_{0\tilde{y}\tilde{y}}\bar{y}_0 - \bar{A}_{0\tilde{y}\tilde{y}}\tilde{y}_0] \quad (C.40)$$

With equation (C.40), one can eliminate the dependency on  $\hat{x}_0$  from all the above state equations. The resulting system state model has the following form

$$\left\{ \begin{aligned} \begin{bmatrix} \dot{\hat{y}}_0 \\ \dot{\tilde{y}}_0 \\ \dot{\tilde{y}}_0 \\ \dot{\tilde{y}}_0 \\ \dot{\tilde{x}}_0 \end{bmatrix} &= \begin{bmatrix} \hat{A}_{0\hat{y}\hat{y}} & \hat{A}_{0\hat{y}\tilde{y}} & \hat{A}_{0\hat{y}\tilde{y}} & \hat{A}_{0\hat{y}\tilde{y}} & \hat{A}_{0\hat{y}\tilde{x}} \\ 0 & 0 & 0 & I & 0 \\ \hat{A}_{0\tilde{y}\hat{y}} & \hat{A}_{0\tilde{y}\tilde{y}} & \hat{A}_{0\tilde{y}\tilde{y}} & 0 & 0 \\ \hat{A}_{0\tilde{y}\hat{y}} & \hat{A}_{0\tilde{y}\tilde{y}} & \hat{A}_{0\tilde{y}\tilde{y}} & \hat{A}_{0\tilde{y}\tilde{y}} & \hat{A}_{0\tilde{y}\tilde{x}} \\ \hat{A}_{0\tilde{x}\hat{y}} & \hat{A}_{0\tilde{x}\tilde{y}} & \hat{A}_{0\tilde{x}\tilde{y}} & \hat{A}_{0\tilde{x}\tilde{y}} & \hat{A}_{0\tilde{x}\tilde{x}} \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ \tilde{y}_0 \\ \tilde{y}_0 \\ \tilde{x}_0 \end{bmatrix} + \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ \hat{B}_{0\tilde{y}} & \tilde{B}_{0\tilde{y}} & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0\tilde{x}} & L_{0\tilde{x}} \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \\ \tilde{y} \end{bmatrix} \\ y &= [\hat{C}_0 \quad \bar{C}_0 \quad \tilde{C}_0 \quad 0 \quad 0] \begin{bmatrix} \hat{y}_0 \\ \tilde{y}_0 \\ \tilde{y}_0 \\ \tilde{y}_0 \\ \tilde{x}_0 \end{bmatrix} \end{aligned} \right. \quad (C.41)$$

Let's apply the following state transformation to eliminate the term  $\hat{B}_{0\tilde{y}}$  in the input matrix associated with the input  $\hat{u}_0$ ,

$$\begin{aligned} \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0\tilde{y}} & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0\tilde{x}} & L_{0\tilde{x}} \end{bmatrix} &= \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ -P & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ \hat{B}_{0\tilde{y}} & \tilde{B}_{0\tilde{y}} & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0\tilde{x}} & L_{0\tilde{x}} \end{bmatrix} \\ &= T_P \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\hat{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ 0 & 0 & L_{0\tilde{y}} \\ \hat{B}_{0\tilde{y}} & \tilde{B}_{0\tilde{y}} & L_{0\tilde{y}} \\ 0 & \tilde{B}_{0\tilde{x}} & L_{0\tilde{x}} \end{bmatrix} \end{aligned} \quad (C.42)$$

where

$$\begin{aligned} P &= \hat{B}_{0\tilde{y}}\hat{\Sigma}_{B_{0y}}^{-1} \\ \tilde{B}_{0\tilde{y}} &= \tilde{B}_{0\tilde{y}} \\ L_{0\tilde{y}} &= -PL_{0\hat{y}} + L_{0\tilde{y}} \end{aligned} \quad (C.43)$$

Applying the same transformation  $T_P$  to the system and output matrices, we obtain

$$\begin{bmatrix} \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{x}} \\ 0 & 0 & 0 & I & 0 \\ \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & 0 & 0 \\ \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{x}} \\ \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{x}} \end{bmatrix} = T_P^{-1} \begin{bmatrix} \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{x}} \\ 0 & 0 & 0 & I & 0 \\ \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & 0 & 0 \\ \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{y}} & \hat{A}_{0\check{y}\check{x}} \\ \hat{A}_{0\check{x}\check{y}} & \hat{A}_{0\check{x}\check{y}} & \hat{A}_{0\check{x}\check{y}} & \hat{A}_{0\check{x}\check{y}} & \hat{A}_{0\check{x}\check{x}} \end{bmatrix} T_P$$

$$[\hat{C}_0 \quad \bar{C}_0 \quad \tilde{C}_0 \quad 0 \quad 0] = [\hat{C}_0 \quad \bar{C}_0 \quad \tilde{C}_0 \quad 0 \quad 0] T_P \quad (C.44)$$

The new system is of the form

$$\begin{cases} \begin{bmatrix} \dot{\hat{y}}_0 \\ \dot{\bar{y}}_0 \\ \dot{\tilde{y}}_0 \\ \dot{\check{y}}_0 \\ \dot{\check{x}}_0 \end{bmatrix} = \begin{bmatrix} \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{x}} \\ 0 & 0 & 0 & I & 0 \\ \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & 0 & 0 \\ \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{y}} & \check{A}_{0\check{y}\check{x}} \\ \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{y}} & \check{A}_{0\check{x}\check{x}} \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \bar{y}_0 \\ \tilde{y}_0 \\ \check{y}_0 \\ \check{x}_0 \end{bmatrix} + \begin{bmatrix} \hat{\Sigma}_{B_{0y}} & 0 & L_{0\check{y}} \\ 0 & 0 & L_{0\check{y}} \\ 0 & 0 & L_{0\check{y}} \\ 0 & \tilde{B}_{0\check{y}} & L_{0\check{y}} \\ 0 & \tilde{B}_{0\check{x}} & L_{0\check{x}} \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \tilde{u}_0 \\ \check{y} \end{bmatrix} \\ y = [\hat{C}_0 \quad \bar{C}_0 \quad \tilde{C}_0 \quad 0 \quad 0] \begin{bmatrix} \hat{y}_0 \\ \bar{y}_0 \\ \tilde{y}_0 \\ \check{y}_0 \\ \check{x}_0 \end{bmatrix} \end{cases} \quad (C.45)$$

The next iteration can be carried out in a similar fashion for the subsystem corresponding to the system states  $\check{y}_0$  and  $\check{x}_0$ . For convenience, we re-define them as  $y_1$  and  $x_1$  respectively,

$$\begin{cases} y_1 = \check{y}_0 \\ x_1 = \check{x}_0 \end{cases} \quad (C.46)$$

and letting the inputs  $\check{y}_1$  and  $u_1$  be

$$\check{y}_1 = \begin{bmatrix} \check{y} \\ \hat{y}_0 \\ \bar{y}_0 \\ \check{y}_0 \end{bmatrix} \quad (C.47)$$

$$u_1 = \tilde{u}_0 \quad (C.48)$$

At the iteration  $k = 1$ , the system now has the familiar form of equation (C.21) corresponding to the case  $k = 0$ ,

$$\begin{bmatrix} \dot{y}_1 \\ \dot{x}_1 \end{bmatrix} = \begin{bmatrix} A_{1yy} & A_{1yx} \\ A_{1xy} & A_{1xx} \end{bmatrix} \begin{bmatrix} y_1 \\ x_1 \end{bmatrix} + \begin{bmatrix} B_{1y} \\ B_{1x} \end{bmatrix} u_1 + \begin{bmatrix} L_{1y} \\ L_{1x} \end{bmatrix} \check{y}_1 \quad (C.49)$$

For the above system, we apply the same procedure as developed for the case of  $k = 0$ . The SCB algorithm continues until the term  $\bar{A}_{k\check{y}\check{x}}$  is equal to zero. At the last  $k^{th}$ -iteration, the subsystem will be of the following form

$$\begin{bmatrix} \dot{\hat{y}}_k \\ \dot{\bar{y}}_k \\ \dot{\tilde{y}}_k \\ \dot{\check{x}}_k \end{bmatrix} = \begin{bmatrix} \bar{A}_{k\hat{y}\hat{y}} & \bar{A}_{k\hat{y}\tilde{y}} & \bar{A}_{k\hat{y}\check{x}} \\ \bar{A}_{k\tilde{y}\hat{y}} & \bar{A}_{k\tilde{y}\tilde{y}} & 0 \\ \bar{A}_{k\check{x}\hat{y}} & \bar{A}_{k\check{x}\tilde{y}} & \bar{A}_{k\check{x}\check{x}} \end{bmatrix} \begin{bmatrix} \hat{y}_k \\ \tilde{y}_k \\ \check{x}_k \end{bmatrix} + \begin{bmatrix} \hat{\Sigma}_{B_{ky}} & 0 & L_{k\hat{y}} \\ 0 & 0 & L_{k\tilde{y}} \\ 0 & \tilde{B}_{k\check{x}} & L_{k\check{x}} \end{bmatrix} \begin{bmatrix} \hat{u}_k \\ \tilde{u}_k \\ \check{y}_k \end{bmatrix} \quad (C.50)$$

In the above recursive algorithm, the system states will have the following components

$$x_{scb} = \begin{bmatrix} \hat{y}_0 \\ \bar{y}_0 \\ \tilde{y}_0 \\ \hat{y}_1 \\ \bar{y}_1 \\ \tilde{y}_1 \\ \vdots \\ \hat{y}_j \\ \bar{y}_j \\ \tilde{y}_j \\ \vdots \\ \hat{y}_k \\ \bar{y}_k \\ \bar{x}_k \end{bmatrix} \quad (C.51)$$

We note the following useful relations

$$\begin{aligned} \dim(\hat{y}_j) &= \dim(\hat{u}_j) \\ \dim(\bar{y}_{j-1}) &= \dim\left(\begin{bmatrix} \hat{y}_j \\ \bar{y}_j \\ \tilde{y}_j \end{bmatrix}\right) \end{aligned} \quad (C.52)$$

for  $0 \leq j \leq k$ .

Finally, we can partition  $x_{scb}$  further into the set of  $y_f$ ,  $y_b$ ,  $x_c$  and  $x_a$  states defined as follows,

$$y_f = \begin{bmatrix} \hat{y}_0 \\ \bar{y}_0 \\ \hat{y}_1 \\ \bar{y}_1 \\ \vdots \\ \hat{y}_j \\ \bar{y}_j \\ \vdots \\ \hat{y}_k \end{bmatrix} \quad (C.53)$$

The number of infinite zeros of order  $j$  is equal to the dimension of the vector  $\hat{y}_j$  (or  $\hat{u}_j$ ) for  $0 \leq j \leq k$ .

$$y_b = \begin{bmatrix} \tilde{y}_0 \\ \tilde{y}_1 \\ \vdots \\ \tilde{y}_j \\ \vdots \\ \tilde{y}_k \end{bmatrix} \quad (C.54)$$

The system structure associated with the state equations for  $\tilde{y}_j$  has no direct input terms from  $\hat{u}_j$ ,  $\bar{u}_k$  and  $\bar{x}_j$  for  $0 \leq j \leq k$ . Hence, the presence of any of these states would indicate that the overall system is not right-invertible.

If the term  $\tilde{B}_{k\bar{x}}$  at the last iteration is nonzero, then the set of inputs  $\tilde{u}_k$  exists and the system will not be left-invertible. In this case, one can further partition the states  $\bar{x}_k$  into  $x_a$  and  $x_c$  states where  $x_c$  belongs to the controllable subspace of  $\tilde{u}_k$ . Namely,

$$\begin{bmatrix} \dot{x}_c \\ \dot{x}_a \end{bmatrix} = \begin{bmatrix} A_{cc} & A_{ca} \\ 0 & A_{aa} \end{bmatrix} \begin{bmatrix} x_c \\ x_a \end{bmatrix} + \begin{bmatrix} 0 & B_c & L_c \\ 0 & 0 & L_a \end{bmatrix} \begin{bmatrix} \hat{u}_k \\ \tilde{u}_k \\ \dot{y}_k \end{bmatrix} \quad (\text{C.55})$$

Eigenvalues of the matrix  $A_{aa}$  corresponding to the uncontrollable subsystem  $x_a$  are simply the system invariant zeros.



## Appendix D

# Robust Gradient Routines

intgl2 .....

- FORTRAN subroutine for computing:  $\mathcal{X} = \int_0^t e^{Fp*s} \text{Cov0} e^{Fp'*s} ds$

- INPUTS:

**Nsysord** is the order of the system matrix

**N2sysord** is the row dimension of Cov0, usually = Nsysord

**Fp**, the system matrix

**Cov0**, the base matrix within the integral (see above formula)

**work**, an double-precision valued work array

**iwork**, an integer valued work array

**tf**, the final time

- OUTPUTS:

$\mathcal{X}$  matrix

- SUBROUTINE CALLS:

**abx** matrix multiplication

**abtx** matrix multiplication of one matrix and the transpose of another

**atbx** matrix multiplication of the transpose of a matrix and another

**abte** transpose of a matrix

**lsolve** solves a system of equations, often used to compute matrix inverses

conex2 .....

- FORTRAN subroutine for computing

$$\mathcal{M} = \int_0^t \int_0^v e^{Fp'*(v-s)} CtQC e^{Fp*v} * Gamp * Gamp' * e^{Fp'*s} ds dv$$

- INPUTS:

**Nsysord** is the order of the system matrix

**N2sysord** is the row dimension of CtQC, usually equal to Nsysord

**Nnoises** is the column dimension of Gamp, or the number of independent noises

**Fp**, the system matrix

**CtQC**, the base matrix within the integral (see above formula)

**Gamp**, the secondary base matrix within the integral (see above formula)

**work**, a double-precision valued work array

**iwork**, an integer valued work array

**tf**, the final time

- OUTPUTS:

**DCost**, the  $\mathcal{M}$  matrix

- SUBROUTINE CALLS:

**abx** matrix multiplication

**abtx** multiplication of one matrix and the transpose of another

**atbx** multiplication of the transpose of one matrix and another

**abte** transpose of a matrix

**lsolve** solves a system of equations, often used to compute matrix inverses

**PadExp** is the Padé matrix exponentiation routine (without squaring or scaling)

## PadExp .....

- FORTRAN function for matrix exponential Padé series specialized to this case of function evaluation—where the matrix has a known structure

$$\exp \left\{ \begin{bmatrix} Fp' & CtQC & 0 \\ 0 & -Fp & e^{Fp \cdot t} Cov0 \\ 0 & 0 & Fp' \end{bmatrix} dT \right\} = \begin{bmatrix} Qn & Rn & Yn \\ 0 & Sn & Un \\ 0 & 0 & Qn \end{bmatrix}$$

- INPUTS:

**Nsysord** is the order of the system matrix

**N2sysord** is the row dimension of CtQC, usually equal to Nsysord

**serlen** is the length of Padé series desired

**iwork**, an integer valued work array

**dT**, the final time (usually a small increment here)

**Fp**, the system matrix

**CtQC**, a base matrix within the  $\mathcal{M}$  integral

**Cov0**, a base matrix within the  $\mathcal{M}$  integral

**Qn,Rn,Sn,Un,Yn** are all numerator summands for exponential series of a matrix 3 times the dimension of Fp (5 submatrices, not 9, because some terms are 0 and others are repeated)

**Qd,Rd,Sd,Ud,Yd** are all denominator subblocks of the exponential series. These are nominally work arrays

**work1,work2** are temporary storage work matrices (same size as Fp)

**work**, a double-precision valued work array

- OUTPUTS:

**Yn** is the  $\mathcal{M}$  matrix for small scaling time. It is part of the matrix exponential

**Rn**, a part of the overall matrix exponential ([1,2] partition above)

**Un**, a part of the overall matrix exponential ([2,3] partition above)

**Sn**, the exponential of the -Fp matrix and part of the overall matrix exponential above

**Qn**, the exponential of the Fp' matrix and part of the overall matrix exponential above

- SUBROUTINE CALLS:

**abx** matrix multiplication

**abtx** multiplication of one matrix and the transpose of another

**atbx** multiplication of the transpose of a matrix and another

**abte** transpose of a matrix

**lsolve** solves a system of equations, often used to compute matrix inverses

testintg .....

- **TESTINTG** is a program for testing the **intgl2** subroutine. This test is by way of comparison with the “diagonal” method for a series of test cases, both with diagonalizable and non-diagonalizable matrices

- **INPUTS:**

**Test inputs** are a series of system matrices put in the program via FORTRAN DATA statements (they are basically internal to the program)

- **OUTPUTS:**

**Test results** that list the input matrices and the results of the computations for both the diagonal and robust routines are displayed directly to the screen

- **SUBROUTINE CALLS:**

**testin** is the actual subroutine that does the tests, but it is part of the same source file, so it is not all that separate

**eigenv** does an eigenvalue–eigenvector decomposition of the system matrix

**abe** assigns one matrix to another

**lsolve** solves systems of equations, in this case it is used to find a matrix inverse

**fexp** is a specialized scalar exponentiation routine particular to **SANDY**

**aatx** multiplies a matrix by its transpose

**abx** multiplies two matrices

**abtx** multiplies one matrix by the transpose of another

**intglx** is the original “diagonal” means of calculating  $\mathcal{X}$

**prntm** prints a matrix out to the screen with a banner and label

**intgl2** is the “robust” subroutine for computing  $\mathcal{X}$

testcone .....

- **TESTCONE** is a program for testing the **conex2** subroutine. This test is by way of comparison with the “diagonal” method for a series of test cases, both with diagonalizable and non-diagonalizable matrices

- **INPUTS:**

**Test inputs** are a series of system matrices put in the program via FORTRAN DATA statements (they are basically internal to the program) These tests are the same as for the program *testintg*

- **OUTPUTS:**

**Test results** that list the input matrices and the results of the computations for both the diagonal and robust routines are displayed directly to the screen

- **SUBROUTINE CALLS:**

**tstrun** is the actual subroutine that does the tests, but it is part of the same source file, so it is not all that separate

**eigenv** does an eigenvalue-eigenvector decomposition of the system matrix

**abe** assigns one matrix to another

**lsolve** solves systems of equations, in this case it is used to find a matrix inverse

**fexp** is a specialized scalar exponentiation routine particular to **SANDY**

**aatx** multiplies a matrix by its transpose

**abx** multiplies two matrices

**abtx** multiplies one matrix by the transpose of another

**atbx** multiplies the transpose of one matrix by the other

**conexp** is the original “diagonal” means of calculating  $\mathcal{M}$

**prntm** prints a matrix out to the screen with a banner and label

**conex2** is the “robust” subroutine for computing  $\mathcal{M}$

**intgl2** is the “robust” subroutine for computing  $\mathcal{X}$



## Appendix E

# General Utility Routines for Synthesis and Analysis

pseudoi.m .....

- $T_{inv} = \text{pseudoi}(T)$ ;  
MATLAB function for computing pseudo-inverse of  $T$  via the Singular Value Decomposition. An inverse of  $T$  is guaranteed in the sense that  $T_{inv} * T = I'$ , where  $I'$  will have either 1's or 0's along the diagonal, and 0's everywhere else. The number of 1's is equal to the rank of  $T$
- INPUTS:  
 $T$  is the input matrix
- OUTPUTS:  
 $T_{inv}$  is the pseudo-inverse

rmodal.m .....

- $[T, D] = \text{rmodal}(A)$ ; MATLAB routine for real eigenvalue/eigenvectors
- INPUTS:  
 $A$  matrix to be decomposed
- OUTPUTS:  
 $T$  real transformation matrix  
 $D$  "Diagonalized" matrix of real eigenvalues and  $2 \times 2$   $\sigma - \omega$  blocks

## balreal2.m .....

- `[Abal,Bbal,Cbal,HSV,T] = balreal2(A,B,C);`  
Robust form for balanced realization: algorithm works even when the system contains uncontrollable/unobservable modes
- INPUTS:  
**A,B,C**, System dynamics, input distribution, and output distribution matrices
- OUTPUTS:  
**Abal,Bbal,Cbal** Balanced state matrices  
**HSV** are the Hankel singular values  
**T** is the transformation matrix for balanced realization

## gainloci.m .....

- `[zeroid] = gainloci(A,B,C,D,K,output,input,gains);`  
MATLAB routine to compute closed loop eigenvalues for field of gains (negative feedback only). Normally used in single loop gain margin computation
- INPUTS:  
**A,B,C,D** System matrices  
**K** Nominal feedback gain (scale factor) for loop to be evaluated  
**output** is the index of the output for the loop to be fed back  
**input** is the index of the input of the loop fed back  
**gains** is the field of gains, relative to  $K$
- OUTPUTS:  
**zeroid** is the closed loop eigenvalue series corresponding to the gain field  
**Gain margin** in both magnitude and db

phasloci.m .....

- `[zeroid] = phasloci(A,B,C,D,K,output,input,phases);` MATLAB routine to compute closed loop eigenvalues for a set of phase values (deg), negative feedback only. Normally used in single-loop gain margin computation

- INPUTS:

**A,B,C,D** System matrices

**K** Nominal feedback gain (scale factor) for loop to be evaluated

**output** is the index of the output for the loop to be fed back

**input** is the index of the input of the loop fed back

**phases** is the set of phases (deg)

- OUTPUTS:

**zeroid** is a series of closed-loop eigenvalues corresponding to the given set of phases

**Phase margin** in degrees

lqrcross.m .....

- `[K,S] = lqrcross(A,B,C,D,Q,R);`  
MATLAB routine to compute state feedback gains and Riccati solution by direct eigenvalue/eigenvector partitioning

- INPUTS:

**A,B,C,D** System matrices, with control input  $u$  in, criterion output  $z$

**Q** Criterion weighting matrix

**R** Control weighting matrix

- OUTPUTS:

**K** State feedback matrix

**S** Riccati solution

maklticl.m .....

- **[Acl,Bcl,Ccl,Dcl] = maklticl(A,B,C,D,Bi,Di,Ac,Bc,Cc,Dc)**  
Creates Linear Time Invariant time domain closed loop system matrices from plant (A,B,C,D,Bi,Di) and controller (Ac,Bc,Cc,Dc) matrices. The plant has both a control input  $u$  and a command input  $U_c$ :

$$\begin{aligned}\frac{dx}{dt} &= A * x + B * u + Bi * U_c \text{ (often } B=Bi, \text{ but not always)} \\ y &= C * x + D * u + Di * U_c\end{aligned}$$

Assuming NEGATIVE feedback from the controller, this routine uses the following equations:

$$\begin{aligned}Acl &= \begin{bmatrix} A - B(I + Dc * D)^{-1} Dc * C & -B(I + Dc * D)^{-1} Cc \\ Bc(I + D * Dc)^{-1} C & Ac - Bc(I + D * Dc)^{-1} D * Cc \end{bmatrix} \\ Bcl &= \begin{bmatrix} Bi - B(I + Dc * D)^{-1} Dc * Di \\ Bc(I + D * Dc)^{-1} D \end{bmatrix} \\ Ccl &= \begin{bmatrix} (I + D * Dc)^{-1} C & -(I + D * Dc)^{-1} D * Cc \\ -(I + Dc * D)^{-1} Dc * C & -(I + Dc * D)^{-1} Cc \end{bmatrix} \\ Dcl &= \begin{bmatrix} (I + D * Dc)^{-1} Di \\ -(I + Dc * D)^{-1} Dc * Di \end{bmatrix}\end{aligned}$$

If either  $(I + D * Dc)$  or  $(I + Dc * D)$  are non-invertible, the solution is beyond the scope of this subroutine

- **INPUTS:**

**A,B,C,D,Bi,Di** are the plant matrices, with Bi and Di being the command input distribution matrices

**Ac,Bc,Cc,Dc** are the controller matrices

- **OUTPUTS:**

**Acl,Bcl,Ccl,Dcl** are the closed-loop output matrices. The columns of Bcl and Dcl correspond to the command input  $U_c$ .

**sandy2.exe** .....

- Executable program for controller design via optimization. This form of **SANDY** utilizes both diagonal and robust forms of the gradient computation algorithm. Switches between algorithms according to the condition number of the eigenvector matrix derived from the system dynamics matrix.
- INPUTS:  
**Data file** for describing the plant, controller design parameters, and current controller setup.
- OUTPUTS:  
**Log file** for the optimization run, reports how well the optimization did and many characteristics of the final closed loop system.  
**SANDY** formatted file for describing current controller  
**MATLAB** formatted file for describing current controller

**wcsandy.exe** .....

- Executable program for controller design via optimization. This form of **SANDY** utilizes the diagonal form of the gradient computation algorithm. This program also uses the variant “worst-case” cost function approach.
- INPUTS:  
**Data file** for describing the plant, controller design parameters, and current controller setup.
- OUTPUTS:  
**Log file** for the optimization run, reports how well the optimization did and many characteristics of the final closed loop system.  
**SANDY** formatted file for describing current controller  
**MATLAB** formatted file for describing current controller



## Appendix F

# UH-60 Rotocraft Design Files (Chapter 7)

### uh60comd.m .....

- MATLAB script file for generating and plotting command responses
- INPUTS: (no formal argument list—items named actually used)
  - Acl** Nomenclature for dynamics matrix
  - Bcmd** Command input distribution matrix
  - Ceval** Output distribution matrix—assume the outputs are in the order:  $p, q, r, u, v, w, \phi, \theta, \delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .
  - Dcmd** Command to output direct distribution
  - icmd** Selects appropriate column of Bcmd
  - U2** Command input profile
  - T2** Time base
- OUTPUTS:
  - Y**, the generated responses from the MATLAB function *lsim*. The plots of  $p, q, r, u, v, w, \phi, \theta, \delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$  are in 3 sets of panels.
  - Plots** of command responses.

### makidea0.m (makidea1.m) .....

- MATLAB script file to fabricate the ideal response model in the system matrices: Aideal, Bideal, Cideal, and Dideal.
- OUTPUTS:
  - Aideal, Bideal, Cideal, Dideal** are the literal matrices (namewise) created—no argument list for input or output. 11th order system.

**ad2delay.m** .....

- `[Ad,Bd,Cd,Dd] = ad2delay(A,B,C,D,Tdelay);`  
MATLAB function to add 4 first-order actuator delays to the system model.
- INPUTS:  
**A,B,C,D** is the system model, assumed with 7 inputs.  
**Tdelay** is the delay time, in seconds.
- OUTPUTS:  
**Ad,Bd,Cd,Dd** is the delayed actuator output system.

**adodelay.m** .....

- `[Ad,Bd,Cd,Dd] = adodelay(A,B,C,D,Tdelay);`  
MATLAB function to add 12 first order sensor delays to the system model.
- INPUTS:  
**A,B,C,D** is the system model, assumed with 31 states and either 12 or more sensor outputs.  
**Tdelay** is the delay time, in seconds.
- OUTPUTS:  
**Ad,Bd,Cd,Dd** is the delayed sensor output system.

**fabsylq0.m (fabsylq1.m)** .....

- MATLAB script file for design of state feedback for the hover (forward flight) flight condition. Used in Luenberger Observer design; includes delayed outputs to integrators.

- INPUTS:

**f01\_g2r2.mat (f15\_g2r2.mat)** UH-60 state model linearized from nonlinear simulation program output. Flight condition point: 1 knot forward velocity (15 knots forward velocity) . Additional environmental parameters associated with this flight condition: air density 0.002377 slug/cubic foot; rotor speed 27 rad/s; and gross weight 16,800 lbs.

**Q and R** weights are internal to the file, nonetheless they are nominal “inputs”.

- OUTPUTS:

**State feedback matrix K** in file: kmatalt0.mat (kmatalt1.mat).

**Plots** of command responses for closed loop design

- CALLS TO:

**makidea0.m (makidea1.m)** produces the ideal response/feedforward.

**ad2delay.m** adds 4 actuator delays to the system model.

**adodelay.m** adds 12 sensor delays to the system model.

**lqrcross.m** solves for state feedback gains via partitioning the Hamiltonian matrix.

**uh60cmd.m** to generate the plots of the command responses

**fabuhlq0.m (fabuhlq1.m)** .....

- MATLAB script file for design of state feedback for the hover (forward flight) flight condition. Used as baseline state feedback design for comparison to output feedback designs.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** UH-60 state model linearized from nonlinear simulation program output. Flight condition point: 1 knot forward velocity (15 knots forward velocity) . Additional environmental parameters associated with this flight condition: air density 0.002377 slug/cubic foot; rotor speed 27 rad/s; and gross weight 16,800 lbs.
  - Q and R** weights are internal to the file, nonetheless they are nominal “inputs”. These are the same as those of *fabsylq0 (fabsylq1)*.
- OUTPUTS:
  - State feedback matrix K** in file: *kmat0.mat (kmat1.mat)*.
  - Plots** of command responses for closed loop design
- CALLS TO:
  - makidea0.m (makidea1.m)** produces the ideal response/feedforward.
  - ad2delay.m** adds 4 actuator delays to the system model.
  - adodelay.m** adds 12 sensor delays to the system model.
  - lqrcross.m** solves for state feedback gains via partitioning the Hamiltonian matrix.
  - uh60cmd.m** to generate the plots of the command responses

**evalsys0.m (evalsys1.m)** .....

- MATLAB script file for generating system matrices used in the ADS-33C evaluation file *evalhovr.m (evalford.m)*.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** UH-60 state model linearized from nonlinear simulation program output.
  - K**, the state feedback matrix from the file *kmatalt0.mat (kmatalt1.mat)*.
- OUTPUTS:
  - Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .
  - Bgust, Dgust** with inputs in the principal gust directions
  - Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.
  - G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .

**fabsylo0.m (fabsylo1.m)** .....

- This MATLAB script file does a Luenberger Observer output feedback controller design via CLTR.
- INPUTS:  
**f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.  
**kmatalt0.mat (kmatalt1.mat)** is the file containing the K matrix from the state feedback solution.
- OUTPUTS:  
**Luenberger Observer** dynamics matrices: [A<sub>lc</sub>,B<sub>lc</sub>,C<sub>lc</sub>,G<sub>lc</sub>,P<sub>lc</sub>].  
**fabsylo0.mat (fabsylo1.mat)** is the file containing these matrices.  
Plots of responses.
- CALLS TO:  
**makidea0.m (makidea1.m)** produces the ideal response/feedforward.  
**ad2delay.m** adds 4 actuator delays to the system model.  
**adodelay.m** adds 12 sensor delays to the system model.  
**scb.m** computes the Special Coordinate Basis (SCB) for the given dynamics.  
**lqrcross.m** solves for state feedback gains via partitioning the Hamiltonian matrix.  
**uh60cmd.m** is a script file for plotting the responses to an input.

**fabrolo0.m (fabrolo1.m)** .....

- MATLAB script file for robust analysis of Luenberger Observer-based controller generated in *fabsylo0 (fabsylo1)*.
- INPUTS:  
**fabsylo0.mat (fabsylo1.mat)** , the data files containing the Luenberger Observer-based controller.
- OUTPUTS:  
**Multiloop** actuator phase/gain margins  
**Single Loop** actuator phase and gain margins  
**Single Loop** sensor phase and gain margins
- CALLS TO:  
**gainloci** computes gain margins for single loop.  
**phasloci** computes phase margins for single loop.

**evalosy0.m (evalosy1.m) .....**

- MATLAB script file to generate system dynamics matrices from Luenberger Observer-based controller for use in *evalhovr.m* (*evalford.m*).
- INPUTS:

**fabsylo0.mat (fabsylo1.mat)** the source data on the Luenberger Observer-based controller.
- OUTPUTS:

**Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .  
**Bgust, Dgust** with inputs in the principal gust directions  
**Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.  
**G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .

**fabsyro0.m (fabsyro1.m) .....**

- MATLAB script file to take the balanced realization of a Luenberger Observer-based controller and look at its command responses. Internal straps allow 10th, 14th, or 25th order controller.
- INPUTS:

**balrdlo0.mat (balrdlo1.mat)** is the data file containing the balanced realization of the Luenberger system [Alc,Blc,Clc,Glc,Plc]  
**f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
- OUTPUTS:

**Plots** of responses.
- CALLS TO:

**makidea0.m (makidea1.m)** produces the ideal response/feedforward.  
**ad2delay.m** adds 4 actuator delays to the system model.  
**adodelay.m** adds 12 sensor delays to the system model.  
**scb.m** computes the Special Coordinate Basis (SCB) for the given dynamics.  
**uh60comd.m** is a script file for plotting the responses to an input.

**fabrro0.m (fabrro1.m)** .....

- MATLAB script file to perform robust analysis of reduced order Luenberger-based controller. Internal straps allow 25th or 14th order.
- INPUTS:
  - balrdlo0.mat (balrdlo1.mat)** , the data files containing the balanced Luenberger Observer-based controller.
  - fabsylo0.mat (fabsylo1.mat)** , the data files containing the Luenberger Observer-based controller.
- OUTPUTS:
  - Multiloop** actuator phase/gain margins
  - Single Loop** actuator phase and gain margins
  - Single Loop** sensor phase and gain margins
- CALLS TO:
  - gainloci** computes gain margins for single loop.
  - phasloci** computes phase margins for single loop.

**evalros0.m (evalros1.m)** .....

- MATLAB script file to generate system dynamics matrices from a balanced and reduced Luenberger Observer-based controller for use in *evalhovr.m* (*evalford.m*). Straps for setting the reduction at 6th, 14th, and 25th order are available.
- INPUTS:
  - fabsylo0.mat (fabsylo1.mat)** the source data on the Luenberger Observer-based controller.
  - balrdlo0.mat (balrdlo1.mat)** the source data for a balanced Luenberger Observer-based controller.
  - sanltre0o.m (sanltre1o.m)** the source files for the 7th order reduced feedforward.
- OUTPUTS:
  - Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .
  - Bgust, Dgust** with inputs in the principal gust directions
  - Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.
  - G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .

**sanltrf0.m (sanltrf1.m)** .....

- MATLAB script file to assemble reference system dynamics and controller (based on a 14th order observer based on a reduced Luenberger observer) set-up for SANDY run. This SANDY run is to minimize the white noise response error (includes shaped white noise) between the full-order Luenberger observer-based design and the lower order controller.

- INPUTS:

**balrdlo0.mat (balrdlo1.mat)** is the data file containing the balanced realization of the Luenberger system [Alc,Blc,Clc,Glc,Plc]

**sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.

**makidea0.m (makidea1.m)** is the script file that produces the ideal closed-loop system response dynamics [Aideal,Bideal,Cideal,Dideal].

- OUTPUTS:

**Plots** of the closed-loop response using the reference controller for verification.

**Plots** of the closed-loop response using the initial guess of the reduced controller for verification.

**sanltrf0.mat (sanltrf1.mat)** Reference and controller matrices in the plant—controller format acceptable to SANDY. The SANDY input file sanltrf0.dat (sanltrf1.dat) was eventually produced from this.

- CALLS TO:

**rmodal.m** real-valued eigenvalues and eigenvectors—complex eigenvalues show up as 2x2 blocks.

**uh60cmd.m** is a script file for plotting the responses to an input.

**sanltrh0.m (sanltrh1.m) .....**

- MATLAB script file to assemble a reference system based on state feedback on the UH-60 as well as open loop UH-60 dynamics. The open-loop dynamics of the reference will be closed by an initial controller guess and the recovery error defined by the difference in response between the state-feedback based system and the 14th order observer-based controller stabilized system given the same disturbance inputs.
- INPUTS:
  - sanltrf0.m (sanltrf1.m)** is the data file containing the initial controller guess (from the model matching run).
  - sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
  - kmatalt0.mat (kmatalt1.mat)** is the file containing the K matrix from the state feedback solution.
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
- OUTPUTS:
  - Plots** of the closed-loop response using the reference controller for verification.
  - Plots** of the closed-loop response using the initial guess of the reduced controller for verification.
  - sanltrh0.mat (sanltrh1.mat)** Reference and controller matrices in the plant—controller format acceptable to SANDY. The SANDY input file sanltrh0.dat (sanltrh1.dat) was eventually produced from this.
- CALLS TO:
  - rmodal.m** real-valued eigenvalues and eigenvectors—complex eigenvalues show up as 2x2 blocks.
  - uh60cmd.m** is a script file for plotting the responses to an input.
  - ad2delay.m** adds 4 actuator delays to the system model.
  - adodelay.m** adds 12 sensor delays to the system model.

**fabrolr0.m (fabrolr1.m) .....**

- MATLAB script file to perform robust analysis of a recovered 14th order controller originally reduced from the Luenberger Observer design. Also strapped for possible use of the numerical CLTR design.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
  - sanltrf0n.m (sanltrfln.m)** which contain the recovered and reduced 14th order controller component (sanltrh0n.m and sanltrh1n.m are used given a strap setting).
  - sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
- OUTPUTS:
  - Multiloop** actuator phase/gain margins
  - Single Loop** actuator phase and gain margins
  - Single Loop** sensor phase and gain margins
- CALLS TO:
  - ad2delay.m** adds 4 actuator delays to the system model.
  - adodelay.m** adds 12 sensor delays to the system model.
  - gainloci.m** computes gain margins for single loop.
  - phasloci.m** computes phase margins for single loop.

**evaltrf0.m (evaltrf1.m)** .....

- MATLAB script file to generate system dynamics matrices from a balanced reduced and recovered 14th order controller based on the Luenberger Observer-based controller. Output for use in *evalhour.m* (*evalford.m*). Does some response plotting to verify that a proper assembly of a closed-loop system has been made.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
  - sanltrf0n.m (sanltrf1n.m)** which contain the recovered and reduced 14th order controller component.
  - sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
- OUTPUTS:
  - Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .
  - Bgust, Dgust** with inputs in the principal gust directions
  - Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.
  - G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .
  - uh60cmd.m** is a script file for plotting the responses to an input.

**evaltrh0.m (evaltrh1.m)** .....

- MATLAB script file to generate system dynamics matrices from a balanced reduced and CLTR recovered 14th order controller based on the Luenberger Observer-based controller. Output for use in *evalhovr.m* (*evalford.m*). Does some response plotting to verify that a proper assembly of a closed-loop system has been made.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
  - sanltrh0n.m (sanltrh1n.m)** which contain the recovered and reduced 14th order controller component.
  - sanltre0o.m (snaltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
- OUTPUTS:
  - Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .
  - Bgust, Dgust** with inputs in the principal gust directions
  - Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.
  - G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .
  - uh60cmd.m** is a script file for plotting the responses to an input.

**san14do0.m (san14do1.m)** .....

- MATLAB script file to assemble plant and controller matrices for a direct optimization run. The controller form is the same as for the recovery cases: 14th order Luenberger observer with integral control and 7th order feedforward dynamics.
- INPUTS:
  - sanltrf0.m (sanltrf1.m)** is the data file containing the initial controller guess (from the model matching run).
  - sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
- OUTPUTS:
  - Plots** of the closed-loop response using the initial guess of the controller for verification.
  - san14do0.mat (san14do1.mat)** Reference and controller matrices in the plant—controller format acceptable to SANDY. The SANDY input file san14do0.dat (san14do1.dat) was eventually produced from this.
- CALLS TO:
  - uh60cmd.m** is a script file for plotting the responses to an input.
  - ad2delay.m** adds 4 actuator delays to the system model.
  - adodelay.m** adds 12 sensor delays to the system model.

**fabrodo0.m (fabrodo1.m) .....**

- MATLAB script file to perform robust analysis of a 14th order controller produced by direct optimization.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
  - san14do0n.m (san14do1n.m)** which contain the optimized 14th order controller component.
  - sanltre0o.m (sanltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
- OUTPUTS:
  - Multiloop** actuator phase/gain margins
  - Single Loop** actuator phase and gain margins
  - Single Loop** sensor phase and gain margins
- CALLS TO:
  - ad2delay.m** adds 4 actuator delays to the system model.
  - adodelay.m** adds 12 sensor delays to the system model.
  - gainloci.m** computes gain margins for single loop.
  - phasloci.m** computes phase margins for single loop.

**eval14do0.m (eval14do1.m)** .....

- MATLAB script file to generate system dynamics matrices from a direct optimized 14th order controller based on the Luenberger Observer-based controller. Output for use in *evalhovr.m* (*evalford.m*). Does some response plotting to verify that a proper assembly of a closed-loop system has been made.
- INPUTS:
  - f01\_g2r2.mat (f15\_g2r2.mat)** contains the UH-60 system model for the appropriate flight condition.
  - san14do0n.m (san14do1n.m)** which contain the recovered and reduced 14th order controller component.
  - sanltre0o.m (snaltre1o.m)** which contain the 7th order reduced and recovered feedforward dynamics.
- OUTPUTS:
  - Aeval, Ceval** with outputs:  $p, q, r, u, v, \dot{z}, \phi, \theta, \psi, x, y, z$ .
  - Bgust, Dgust** with inputs in the principal gust directions
  - Bcmd, Dcmd** with input terms in the order: pitch, roll, yaw rate, heave, yaw, altitude.
  - G, D** with standard control inputs:  $\delta_0, \delta_S, \delta_C$ , and  $\delta_{TR}$ .
  - uh60cmd.m** is a script file for plotting the responses to an input.

**evalhovr.m (evalford.m)** .....

- MATLAB script file for evaluating system matrices for the ADS-33C hover (forward flight) requirements.

- INPUTS:

**Output** from appropriate evaluation setup m-file (these files start with the word "eval").

**Test** parameters contained within the evaluation file

- OUTPUTS:

**Test** evaluations for many of the ADS-33C requirements

- SUPPORT ROUTINES: Because the number of routines used in these files is large and they are specific to them, they are listed here. Details of each will follow this list.

**bandwidth** determines bandwidth from system matrices in a manner similar to, yet unlike classical methods.

**peakmmax** evaluates transient and steady-state behavior of output trace (Y).

**passit** evaluates a point to see if it is to the right of a given curve.

**get1stor** does an optimization of a given curve trace to a first order response by adjusting the magnitude, delay and time constant.

**get1stda** returns the RMS error between a given curve trace and a trial first order fit.

**pulshold** checks compliance to ADS-33C 3.2.6 by analyzing response to an actuator pulse (hover only).

**fig5\_3\_4** does a test based on whether a criterion (either bank angle oscillations (Fig 5 of ADS-33C), sideslip excursion (Fig 6 of ADS-33C), or a sort of scaled sideslip excursion (Fig 7 of ADS-33C) is acceptable.

**fig9\_3\_4** checks damping ratio and stability criteria given a sigma-omega pair.

**text934** Reports in text the results from the evaluation done in fig9\_3\_4.

## **bandwidth.m** .....

- **[W180,Wgain,Wphase,Tphase] = bandwidth(Acl,Bcmd,Ccmd,Dcmd,dispflag)**  
MATLAB function to determine bandwidth compatible with definition in mil standard ADS-33C, Figure 2(3.3). This function requires the closed-loop system matrix, the relevant command input, and the relevant controlled output (only 1 each). A Bode plot of the system gain and phase of X/Xcmd is made. The frequency at the phase=180 point is found. Wphase is the frequency at a margin of 45 deg. from that. Wgain is the frequency 6dB higher than the magnitude at W180.
- **INPUTS:**  
**Acl,Bcmd,Ccmd,Dcmd** are the system matrices for bandwidth determination. The loop is closed, and only one input, one output.  
**dispflag** controls verifying the results with a plot the user can see with the 180 frequency and the phase and gain frequencies marked.
- **OUTPUTS:**  
**W180** is the frequency at which the phase is 180 degrees  
**Wgain** is the frequency at which the magnitude is 6db up from that at W180.  
**Wphase** is the frequency at which the phase is 135 degrees  
**Tphase** is known as the phase delay, which is the slope of the phase curve between W180 and the frequency where the phase lags an additional 180 degrees.

## **peakmmax.m** .....

- **[Ymax,Tmax,Ymin,Tmin,Ymax2,Tmax2,Yavg] = peakmmax(Ytrace)**  
Evaluates transient and steady behavior of trace. Trace assumed to be step response with analysis of overshoot and undershoot. Tries (presently without sophistication) to estimate steady state value of the trace. With all of this, the standard functions `max` and `find` need alternate means. With all of this, this function does not ever try to extrapolate to find Yavg, since that can get one into severe trouble. Tmax and Tmin are indices into Ytrace where the respective max and min values are located.
- **INPUTS:**  
**Ytrace** is the step response.
- **OUTPUTS:**  
**Ymax** is the peak maximum.  
**Tmax** is the index into Ytrace where the value is assumed.  
**Ymin** is the first minimum after Ymax.  
**Tmin** is the index into Ytrace where the value Ymin is assumed.  
**Ymax2** is the next local maximum after Ymin.  
**Tmax2** is the index into Ytrace where the value is Ymax2 assumed.  
**Yavg** is the estimated steady-state value.

## **passit.m** .....

- **status = passit(curve,point)**  
This function tests to see if a point is to the right of the given curve. If so *status* is 1, if not, *status* is 0. In *curve* and *point*, the x coordinate is first, then y. The curve is assumed non-closed. Linear extrapolations are done before the begin point and after the end points based on the segments there.
- **INPUTS:**  
  
**curve** is a set of points defining a curve. This curve could be closed, but in general it is assumed non-closed.  
  
**point** is the x,y coordinate of a point to be tested.
- **OUTPUTS:**  
  
**status** is 1 if the point is to the right of the given curve, 0 if it is not.

## **pulshold.m** .....

- **quepasa = pulshold(Ttrace,Ytrace)**  
Checks compliance with 3.2.6 of ADS-33C (Handling Qualities for Military Rotocraft). Ytrace is a response to an actuator pulse. To pass (0 return), it is to rise, then fall to less than 10%peak value within 10 seconds (hence the time trace needed too). It is to stay below this 10% for another 30 seconds. UCE=1 needs 10% settle within only 20 seconds, this is checked too if the original settle is not met. The return condition will be -1 here. It is best to supply around 60 seconds of data (only 50 absolutely necessary). Failure is a return condition of 1.
- **INPUTS:**  
  
**Ytrace** is the response to an actuator pulse.  
**Ttrace** is the corresponding time trace.
- **OUTPUTS:**  
  
**quepasa** is set to 0 if the trace passes the requirement, 1 if it does not.

## **get1stor.m** .....

- **[Mag,Delay,Tconst,corrsq] = get1stor( Times, Ydata )**  
Does first order fit  $y(t+\text{Delay}) = \text{Mag} \cdot (1 - \exp(-t/\text{Tconst}))$  to given data. Computes a sort of chi-squared sense that indicates how well the fit actually behaved. This procedure is presented in Figure 8(3.3) of ADS-33C. This routine calls the MATLAB function *nelder* which will in turn call *get1stda*, which generates data for the optimization routine. The common block: **global ydata Tset Nset** must be present in the root m-file.
- **INPUTS:**  
**Ydata** is the function trace.  
**Times** is the corresponding time trace.
- **OUTPUTS:**  
**Mag** is the magnitude of the first order function.  
**Delay** is the time delay to the nominal zero point for this first-order function.  
**Tconst** is the time constant.  
**corrsq** is the ratio of the squares of the fitted curve and the raw data about the mean.

## **get1stda.m** .....

- **error = get1stda( params )**  
Returns error from latest set of first order function parameter fit. Parameters are: Mag, time delay, and time constant, in that order for the params vector The common block: **global ydata Tset Nset** must be present in the root m-file.
- **INPUTS:**  
**params** is a 3-element vector whose meanings are: Magnitude of first-order function, time delay, and time constant.
- **OUTPUTS:**  
**error** is the RMS error between the data and the fitting function.

▼

## **nelder.m** .....

- `[x, iterats] = nelder(errorfn,parmlist,tol,prnt)`  
*nelder* is a nonlinear optimization function (the Nelder–Mead Simplex algorithm) that tries to adjust the given parameter vector in order to minimize the output of the argument function *errorfn*. This is often a library function within MATLAB, but not always, hence its inclusion here.
- INPUTS:  
**errorfn** is a user-defined function which uses the *parmlist* entry in the *nelder* command line to return the associated value of the cost function. The actual argument entry is the *name* of this function's m-file (which *should* be the name of the function as well).  
**parmlist** is the parameter list (array). The actual meanings of the individual elements depends on your *errorfn* as *nelder* derives the relationships of the individual terms to the cost.  
**tol** is the stopping tolerance (defaults to 1.0e-3).  
**prnt** will describe each step if set to a nonzero value (defaults to 0).
- OUTPUTS:  
**iterats** is the number of iterations (this argument is optional).  
**x** is the optimized parameter list (same length as input).

## **fig5\_3\_4.m** .....

- `[passit] = fig5_3_4(RR2betaP,RollLag,PhOSCAV,LineSpec)`  
Uses function *passit* to evaluate input criteria and outputs what level performance they represent as per Figure 5(3.4) of the ADS-33C rotocraft handling specs. Also evaluate as per Figure 6(3.4) and Figure 7(3.4) given different values of *LineSpec* (the raw criterion data). Will detect whether one has one curve or two (4 cols versus 2). For use in forward flight tests only.
- INPUTS:  
**RR2betaP** is nominally the roll rate to sideslip phase angle.  
**RollLag** is the component of an oscillation cycle needed to reach the first maximum of an overshoot response in the lateral-directional oscillation.  
**PhOSCAV** is either the roll oscillation envelope size or the sideslip oscillation peak normalized by the roll oscillation peak.  
**LineSpec** is the specific test characteristic line using these criteria.
- OUTPUTS:  
**passit** is set to 1, 2, or 3, according to what criteria are met.

**fig9\_3\_4.m** .....

- `[result] = fig9_3_4( sigma, omega )`  
returns 1 if meets the all MTE specs on sigma and omega according to Figure 9(3.4) of ADS-33C, which examines known lateral-directional modes. Returns 2 for Level I (all MTE but slalom, ground attack and air combat), Level II for these. Returns 3 for Level III for slalom, ground attack, and air combat, Level II for all others, Returns 4 for Level III all around, and 5 for unstable. Used in forward flight analysis only.
- INPUTS:  
**sigma,omega** are the real and complex parts of the eigenvalue under consideration.
- OUTPUTS:  
**result** is a simple number from 1 to 5, according to the above description.

**text934.m** .....

- `text934( result )`  
Displays appropriate text string according to number handed it. number same series as in the fig9\_3\_4 evaluation routine. Used in forward flight analysis only.
- INPUTS:  
**result** is the result from the *fig9\_3\_4* routine.
- OUTPUTS:  
**Text** to screen to describe result.



## Appendix G

# UH-60 Rotocraft Models (Chapter 7)

f01\_g2r2.mat (Hover) .....

- 31 state model linearized from nonlinear simulation program output

$p, q, r$	Body axis attitude rates (deg/s)
$u, v, w$	Body axis velocities (ft/s)
$\theta_x, \theta_y, \theta_z$	Euler angles (deg) from terrestrial to rotocraft axes (pitch, roll, yaw)
$x, y, z$	Inertial positions (ft)
$\beta_0, \beta_D, \beta_{1S}, \beta_{1C}$	Rotor flapping rates (deg/s)
$\zeta_0, \zeta_D, \zeta_{1S}, \zeta_{1C}$	Rotor lead-lag rates (deg/s)
$\beta_0, \beta_D, \beta_{1S}, \beta_{1C}$	Rotor flapping angles (deg)
$\zeta_0, \zeta_D, \zeta_{1S}, \zeta_{1C}$	Rotor lead-lag angles (deg)
$\lambda_0, \lambda_{1S}, \lambda_{1C}$	Inflow velocities (1/sec) These are normalized by the rotor radius (26.83ft)

- Flight condition point: 1 knot forward velocity. Additional environmental parameters associated with this flight condition: air density 0.002377 slug/cubic foot; rotor speed 27 rad/s; and gross weight 16,800 lbs
- Dynamics matrix **F**
- Control and disturbance input distribution **G**
- CONTROL INPUTS (deg):  $\delta_0$  (collective),  $\delta_S$  (sine),  $\delta_C$  (cosine), and  $\delta_{TR}$  (tail rotor)
- DISTURBANCE INPUTS (ft/s):  $V_{gx}$ ,  $V_{gy}$ , and  $V_{gz}$ , gusts from the principal terrestrial frame directions

f15\_g2r2.mat (Forward Flight) .....

- 31 state model linearized from nonlinear simulation program output
- Flight condition point: 15 knot forward velocity. Additional environmental parameters associated with this flight condition: air density 0.002377 slug/cubic foot; rotor speed 27 rad/s; and gross weight 16,800 lbs
- System dynamics matrix **F**
- Control and disturbance input distribution **G**
- All other details same as for hover model.

## Appendix H

# JPL Large Space Structure Design Files (Chapter 4)

- lscl1sim.m** .....  
• MATLAB script file for running closed and open loop system dynamics through stability test. Runs stability test mentioned in Chapter 4, the structure is pulsed for 6.4 seconds, then allowed to free oscillate. Also gives eigenvalues of closed loop system.
- INPUTS:  
**lsclnew.mat** is the data file for the open loop antenna structure.  
**lscl4m5c.m** is the **SANDY** controller output file normally used. Its contents have been copied into *lscl1sim.m*.
- OUTPUTS:  
**Eigenvalues** of the closed loop system  
**Plots** of the rundown responses at the two hub sensors of the antenna for both open and closed loop.
- lscl3m5.dat** .....  
**SANDY** input file that eventually produced final controller. Contains weights, disturbance profile, and starting condition details.



## Appendix I

# JPL Large Space Structure Model (Chapter 4)

lsclnew.mat .....

- MATLAB binary data file for linearized and reduced antenna model
- System matrices **a**, **b**, **c**, **d**
- 20th order model; all modes stable, most modes lightly damped (damping 0.007 to 0.01)
- 6 inputs: *HA1* and *HA10* for hub actuators; *RA1*, *RA4*, *RA7* and *RA10* for rib root actuators
- 6 outputs: *HS1* and *HS10* for hub sensors; *RS1*, *RS4*, *RS7* and *RS10* for rib root sensors
- All sensors collocated with corresponding actuators
- All actuator inputs in Newton-meters
- All sensor outputs in radians



## Appendix J

# One-Dimensional Rotocraft Design Files (Chapter 2.7.1)

**helicm0.dat, helicm1.dat, helicm2.dat, helicm3.dat** .....

**SANDY** input files to run the 4 cases of initial condition versus gust response weighting. Weights of 0, 0.1, 1.0 and 10. on the initial conditions are used.

**helilr0.dat, helilr1.dat, helilr2.dat, helilr3.dat** .....

**SANDY** input files to run the 4 cases of loop recovery where the actuator reliance is lessened by injecting fictitious noise there. Weights of 0, 0.1, 1.0, and 10. are used for this actuator noise. Files are input files for both regular ( $H^2$ ) and worst-case versions of **SANDY**.

**helicm.m** .....

- **MATLAB** script file for analyzing  $H^2$  versus worst-case algorithm performance for a series of designs involving increasing degrees of stability augmentation over disturbance rejection.

- **INPUTS:**

**heli1d.mat** is the **MATLAB**-compatible data file for the basic open-loop one-dimensional helicopter model.

**helicm0h.m, helicm1h.m, helicm2h.m, helicm3h.m** are the  $H^2$  design output files from their respective **SANDY** runs.

**helicm0w.m, helicm1w.m, helicm2w.m, helicm3w.m** are the worst-case design output files from their respective **WCSANDY** runs.

- **OUTPUTS:**

**RMS** values of disturbance responses for displacement and pitch

**Gain** margins of the closed loop system for the 8 cases.

**Phase** margins of the closed loop system for the 8 cases.

**Plots** of the command responses to a position ( $X$ ) change.

**helilr.m** .....

- MATLAB script file for analyzing  $H^2$  versus worst-case algorithm performance for a series of designs involving increasing degrees of actuator-oriented loop recovery over disturbance rejection.

- INPUTS:

**heli1d.mat** is the MATLAB-compatible data file for the basic open-loop one-dimensional helicopter model.

**helilr0h.m, helilr1h.m, helilr2h.m, helilr3h.m** are the  $H^2$  design output files from their respective **SANDY** runs.

**helilr0w.m, helilr1w.m, helilr2w.m, helilr3w.m** are the worst-case design output files from their respective **WCSANDY** runs.

- OUTPUTS:

**RMS** values of disturbance responses for displacement and pitch

**Gain** margins of the closed loop system for the 8 cases.

**Phase** margins of the closed loop system for the 8 cases.

**Plots** of the command responses to a position ( $X$ ) change.